Benchmarking Deep Reinforcement Learning for Battery Dispatch Optimisation

Jerry R. Q. Schonenberg, Kapil Mathur, *Member, IEEE*, Carlos Ros Perez, Detlef Hohl, *Member, IEEE*, Aske Plaat, Thomas M. Moerland, Christian Michler

***Abstract*—With the ongoing energy transition, there is a push for the electrification of various appliances and the use of Behind-the-Meter (BtM) smart batteries. In combination with photovoltaic panels, the carbon footprint and household expenses can be reduced by intelligently managing the power flow around the battery. This can be formulated as a sequential decision making process, where the energy consumption of a residential household must be minimised by the Home Energy Management System (HEMS) by intelligently charging and discharging the BtM battery. Recently, reinforcement learning has been proposed as an alternative to classical approaches to model the HEMS. In this work, we have conducted an extensive comparison between various optimisation algorithms; predominantly from the reinforcement learning paradigm, but also Mixed-Integer Linear Programming (MILP) with perfect foresight and an expert system. In addition, we propose an extension to the Deep Q-Network algorithm by incorporating shared state representation learning over two ensembles, which we refer to as Multi Dynamics- and Q-Learning (MDQL). We empirically demonstrate that MDQL outperforms all other approaches by a significant margin, with the exception of MILP. An in-depth behaviour analysis shows that there are still gains to be made by MDQL in terms of grid tariff exploitation.**

***Index Terms*—Battery Dispatch Optimisation, Deep Reinforcement Learning, Home Energy Management Systems, Mixed-Integer Linear Programming.**

I. Introduction

In order to limit the rise of the global temperature, electricity generation from renewable power sources (e.g., solar, hydro and wind) will have to be scaled up significantly. However, this increase poses as a major challenge for the power grid, which is often already operating at its maximum capacity. Subsequently, overloading the grid will occur more frequently if no further investments are made into the infrastructure. By investing in smart Behind-the-Meter (BtM) batteries, each building will be able to locally store its surplus of self-generated power through photovoltaic (PV) panels, instead of netting it back into the power grid. Consequently, the load on the grid will be decreased, and the building reduces its consumption costs and carbon footprint by maximising its utilisation of PV-power.

In this work, we consider a residential household equipped with PV-panels, BtM battery and a heatpump as its only appliance. Given this, a Home Energy Management System (HEMS) [1] is able to control the power flow around the BtM battery. At each timestep, the HEMS can select the operation of the battery; charge, discharge or remain idle. This poses an optimisation problem, where we want to intelligently manage the power flow such that the PV utilisation and consumption costs are maximised and minimised, respectively. Moreover, we can model this as a sequential decision making problem.

There is a gap in the current literature regarding extensively comparing expert systems and general optimisation approaches, such as reinforcement learning (RL). Moreover, the HEMS paradigm lacks a widely accepted problem setting for benchmarking [2], resulting in the fact that work can not be compared directly with each other. Altogether, this research space lacks work that extensively benchmarks algorithms for HEMS.

In addition, RL algorithms tend to be unstable in this problem setting, for which we propose a novel extension to the Deep Q-Network [3] algorithm, which resolves its instability and can function as the foundation for future work.

Following this, the main contributions of the paper are three-fold:

- Providing an extensive benchmark study, with the emphasis on algorithms from the reinforcement learning (RL) paradigm. For context, Mixed-Integer Linear Programming (MILP) with perfect foresight and a Heuristic-Based System are also evaluated on this problem setting.
- Introducing a novel extension to the Deep Q-Network [3] algorithm, where we utilise shared state representation learning in order to improve its performance, stability and sample efficiency. We refer to this as Multi Dynamics- and Q-Learning (MDQL).
- A novel take on the HEMS problem setting, in the form of an environment that aims to accurately model a residential household located in The Hague.

By means of experimentation, we demonstrate that MDQL is able to outperform all other algorithms, with the exception of MILP with perfect foresight. A behaviour analysis of MDQL shows that there are still gains to be made in terms of grid tariff exploitation.

The remainder of the paper is structured as follows: Section II contains related work; Section III discusses the required preliminaries; Section IV describes the problem setting; Section V contains all optimisation algorithms that are

J. Schonenberg, A. Plaat and T. Moerland are with the Leiden Institute of Advanced Computer Science, Leiden University, Leiden, The Netherlands.

K. Mathur, C. Ros Perez, D. Hohl, and C. Michler are with Shell Global Solutions International B.V., located in Bangalore, India, and London, United Kingdom, and Boston, United States, and Amsterdam, The Netherlands, respectively.

included in this benchmark study; Section VI and Section VII contain the experimental setup and results, respectively; and lastly, Section VIII discusses the main takeaways along with future work suggestions.

II. Related Work

HOME Energy Management Systems (HEMS) have become an increasingly relevant research topic due to the energy transition. The main task is to efficiently manage the energy flow within a house/building, for instance through managing the charge and discharge cycles of a smart battery, Heating Ventilation and Air Conditioning (HVAC) management, scheduling household devices and/or predicting future power demands based on historical data.

For instance, (Mixed-Integer) Linear Programming (LP) [4], [5] solvers have been proposed to optimise the scheduling of appliances with a set of defined constraints (e.g., an appliance must run within the next 12 hours, the smart battery can not exceed a State of Charge of 13 kWh, etc.). However, these solvers scale exponentially with the horizon, which makes the method computationally expensive and quickly intractable. In addition, a (near-)perfect foresight or accurate forecasts over the horizon is a prerequisite, which makes it less suitable for deployment in the real world. Instead, LP can quantify an upper-bound on the performance and function as a baseline for other optimisation approaches.

Other work proposes the use of evolutionary algorithms to optimise the scheduling. For instance, Hu and Xiao [6] make use of genetic programming to deal with demand response. A population of candidates can be altered by repeatedly applying mutation, cross-over and selection to the population. With every generation, the population converges closer to an optimum in the search space. The selection after each generation is based on a hand-crafted fitness-rule (i.e., objective function) that incorporates the user-comfort and costs. Similarly, particle swarm optimisation has also been applied to this domain, for instance by Lugo-Cordero et al. [7].

Lastly, machine learning techniques have also been proposed as HEMS. For instance, deep neural networks, which are optimised by a genetic algorithm, have been applied to managing the power-flow [8], appliance scheduling [9] or lighting control [10], [11] in a residential house.

Moreover, reinforcement learning [12] has also been applied to energy management within a household. Various papers have been dedicated to HVAC-management optimisation [13], [14], appliance scheduling [15], [16] and power generation optimisation [17], [18]. A survey by Vázquez-Canteli and Nagy [2] provides an elaborate overview on the work that has been done in the HEMS-domain with reinforcement learning. The authors make the observation that it is difficult to compare the algorithms, due to the varying nature of the problem settings that are being solved over the papers. There is no mainstream benchmark available to evaluate a RL-agent against.

III. Background

A sequential decision making problem can be formalised into a Markov Decision Process (MDP) [19] $\mathcal{M} \doteq \{\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, p(s_0), \gamma\}$, where $\mathcal{S}$ defines the state space ($s \in \mathcal{S}$); $\mathcal{A}$ is the set of actions ($a \in \mathcal{A}$); $\mathcal{T}$ is the transition function which provides a mapping from state-action pairs to states, i.e.: $\mathcal{T} : \mathcal{S} \times \mathcal{A} \mapsto p(\mathcal{S})$; $\mathcal{R}$ is the reward function, mapping a transition to a numerical value: $\mathcal{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto \mathbb{R}$; $p(s_0)$ is the probability distribution of the initial state of an episode; and γ is the discount factor.

Given the MDP, the objective of the agent is to maximise its discounted cumulative reward $G_{t:t+k}$, denoted in (1), from timestep t to $t+k$ on-wards, and doing so will result in solving the problem that the environment represents.

$$\begin{aligned} G_{t:t+k} &\doteq r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + ... + \gamma^k r_{t+k} \\ &= \sum_{n=0}^{k} \gamma^n r_{t+n} \end{aligned} \tag{1}$$

The main idea is that the RL algorithm will estimate $G_{t:t+k}$ for a given state or state-action pair. These estimators are referred to as value functions v and q, respectively. The strategy of selecting the next action based upon the estimators is referred to as the policy π of the agent. The value functions are defined in (2).

$$\begin{aligned} v_\pi(s) &\doteq \mathbb{E}_{\pi,\mathcal{T}}\big[G \,|\, s\big] \\ &= \mathbb{E}_{a\sim\pi(\cdot\,|\,s),\, s'\sim\mathcal{T}(\cdot\,|\,s,a)}\Big[\mathcal{R}(s,a,s') + \gamma v_\pi(s')\Big] \\ q_\pi(s,a) &\doteq \mathbb{E}_{\pi,\mathcal{T}}\big[G \,|\, s,a\big] \\ &= \mathbb{E}_{s'\sim\mathcal{T}(s,a)}\Big[\mathcal{R}(s,a,s') + \gamma \mathbb{E}_{a'\sim\pi(\cdot\,|\,s')}\Big[q_\pi(s',a')\Big]\Big] \end{aligned} \tag{2}$$

Given this, the objective of the algorithm is to obtain the optimal policy π_*, which makes use of the q- and/or v-value functions.

IV. Problem Setting

WITH the preliminaries discussed, we can now define the outline of the environment with which we model the battery dispatch optimisation setting. We consider a residential household equipped with a smart BtM battery, PV panels, access to the power grid, and a heatpump as its sole appliance. The objective is to provide cheap and green power to the heatpump by intelligently managing the battery power flow. Moreover, the environment models Shell's EcoGenie house which is located in The Hague, and makes use of historical data for the weather (forecasts) and grid tariff features. We now discuss the problem setting in its MDP-form, covering the state space, action space and reward function.

A. State Space

The state space $\mathcal{S}$ consists out of features describing the time, weather (forecasts), grid tariff, battery state of charge (SoC), and heatpump status. The environment is partially based on historical data, namely the weather and pricing features are sampled from a dataset. Specifically, day-ahead market prices have been obtained from ANWB Energy[1], weather features and forecasts have been sampled from the National Solar Radiation Database (NSRDB)[2] and the ERA5-Land dataset [20], respectively. For these features holds that we have one trace available, and subsequently an entry of the dataset will always be followed by the same next entry. This means that we are not dealing with a traditional online environment, since it contains on offline component in the form of these features. Subsequently, epistemic uncertainty might play a significant role in this setting since an algorithm will not be able to continuously sample new transitions from the environment.

B. Action Space

The action space $\mathcal{A}$ has been discretised. It consists out of five actions, where each action corresponds to an operation of the smart battery. Below are the five possible actions denoted that $a \in \mathcal{A}$ can take:

- `IDLE`: Do not charge nor discharge the battery.
- `CHARGE_GRID`: Charge the battery from the power grid.
- `CHARGE_PV`: Charge the battery with the PV-panels.
- `DISCHARGE`: Deploy battery-power to the heatpump.
- `NETTING`: Sell battery-energy to the power supplier.

By limiting $\mathcal{A}$ to these five discrete actions, the sample efficiency of the agent will be improved at the cost of not being able to fully control the (dis)charge rates of the battery. However, based upon domain knowledge we argue that operating below the maximum possible rate is only beneficial under very specific and rare circumstances.

Note that concurrently charging and discharging (i.e., float charging) is not available, since it severly deteriorates the battery's state of health (SoH) [21]. In case `DISCHARGE` is not selected, the heatpump power requirements will be met by drawing power from the grid. This is to ensure that the user comfort level is maintained at all times. Important to note is that the heatpump can only receive power from one source at a time, which means that if the action `DISCHARGE` is selected despite the battery having insufficient power to fullfill the requested power, the power will instead be entirely drawn from the grid and the effect of the `DISCHARGE` action will be nullified.

C. Reward Function

The reward function $\mathcal{R}$ consists out of two compnents: (i) the costs c made over the previous timestep; and, (ii) the deterioration of the battery SoH as a result of performing the selected action. This deterioration is approximated by considering the total number of discharge cycles it has endured and the rated number of discharge cycles set by the manufacturer[3]. The reward function is denoted in (3), where λ_{SoH} is an importance scalar for the SoH, κ the cost per percent of SoH, and $\lambda_{w,t}$ a scalar for the overall reward. The cost per percent of SoH is computed by dividing the rated number of discharge cycles with the cost of purchase.

$$\begin{aligned} r_t &\doteq \lambda_{w,t}\big(-c_t - \kappa \cdot \lambda_{\text{SoH}} \cdot \text{SoH}_{\Delta}\big) \\ \text{SoH}_{\Delta} &\doteq \text{SoH}_{t-1} - \text{SoH}_t \end{aligned} \tag{3}$$

The scalar $\lambda_{w,t}$ is added to the reward function to penalise the agent when an action was selected that did not change the battery SoC, thus wasting an action. Its value at a given timestep t is determined based on (4).

$$\lambda_{w,t} = \begin{cases} m, & \text{if } a_t \in \mathcal{A} \setminus \{\texttt{IDLE}\} \text{ and } \text{SoC}_{\Delta} = 0 \\ 1, & \text{otherwise} \end{cases} \tag{4}$$

V. Methods

We now turn our attention to the optimisation algorithms that are part of the benchmark study. First, we discuss an expert system in the form of a set of heuristics. Next, the Mixed-Integer Linear Programming (MILP) solver is briefly discussed, and lastly, we summarise the reinforcement learning approaches. The expert system and MILP-solver serve as baselines and provide context to how well the reinforcement learning approaches perform. In addition, a lower-bound on the performance is also quantified in the form of the 'Idle'-baseline. This baseline demonstrates what the performance in the environment would have been, given that there are no smart BtM battery nor PV-panels present.

A. Heuristic-Based System

The Heuristic-Based System (HBS) is modelled as a chain of six handcrafted if-statements. These if-statements contain thresholds regarding PV-production, battery SoC and/or grid tariff. Moreover, the thresholds are modelled as parameters which have been optimised through Bayesian optimisation.

The HBS first considers charging the battery with PV-power, after which it considers discharging its power into the heatpump, such that the power consumption of the household will be minimised. If the current environment state is not beneficial for charging from PV nor discharging to the heatpump, only then does the HBS consider charging/discharging the battery from/to the power grid in case it is beneficial given the current tariff. If this is also not the case, then the `IDLE`-action is selected.

B. Mixed-Integer Linear Programming

Next, we developed a Mixed-Integer Linear Programming (MILP) solver that is able to determine the optimal sequence of actions over a horizon, which results in the lowest power consumption costs. The solver receives the perfect information of the next h timesteps, which contains the battery SoC, PV-production, grid tariff and heatpump power requirements. Providing perfect foresight is not feasible in practice, and this baseline is instead aimed at quantifying an *upper-bound* on the performance that can be achieved in this setting. In order to let the program take the near future into consideration during the decision making process, we introduce the execution horizon

[1]The day-ahead market prices can be found here: `https://energie.anwb.nl/actuele-tarieven`.

[2]The data viewer of NSRDB can be accessed here: `https://nsrdb.nrel.gov/data-viewer`.

[3]As the battery model, we used the sonnenBatterie10 as a reference. Its specifications can be found here: `https://sonnenbatterie.co.uk/products/sonnenbatterie-10/`.

e. With this, the program still solves the first h timesteps, but only the first e actions are executed in the environment. Thus, the first $h - e$ states of the next horizon will be equal to the last $h - e$ states of the previous horizon.

Due to the nature of MILP, where the decision variables scale exponentially with the horizon h, it is infeasible to solve horizons of multiple weeks or months at once. Subsequently, MILP quantifies an upper bound that is tractable given the time and compute power constraints. In the end, h and e are set to 12 and 6, respectively.

C. Reinforcement Learning

Lastly, we have benchmarked a set of model-free reinforcement learning (RL) algorithms from differing domains. Below, we give an overview of the collection of algorithms selected for the benchmark, along with our novel extension called Multi Dynamics- and Q-Learning (MDQL).

Deep Q-Network (DQN) [3]: Online off-policy RL approach that approximates the q-values with a function approximator Q. This algorithm has been included since it is one of the building blocks of MDQL. It computes the TD-error δ according to (5), in which a target network is used to compute the target q-values. Here, Q is parameterised by the weights θ or θ' for the online and target network, respectively.

$$\delta \doteq r + \gamma \max_{a'} Q_{\theta'}(s', a') - Q_\theta(s, a) \tag{5}$$

Batch Constrained Q-learning (BCQ) [22], [23]: Offline off-policy RL algorithm that restricts the policy to the action distribution G_ω present in the offline dataset. The discrete version of BCQ largely follows DQN, with the main difference being the action selection strategy, where DQN selects the action greedily. In contrast, BCQ considers the probability of the behaviour policy derived from the offline dataset, see (6). The probability of a' is scaled by the action $\hat{a}$ that has the highest probability for s'. Given this, only the subset of actions that exceed threshold Φ_{BCQ} are considered when taking the (arg)max. Note that setting τ to 0 or 1 results in Q-learning or behaviour cloning, respectively.

$$\frac{a' \mid G_\omega(a' \mid s')}{\max_{\hat{a}} G_\omega(\hat{a} \mid s')} > \Phi_{\text{BCQ}} \tag{6}$$

However, since we are dealing with a semi-offline environment (i.e., only a subset of the features are sampled from the offline dataset), the agent still has to collect its transitions. Subsequently, we have made some minor adjustments to the algorithm; we added a replay buffer and ϵ-greedy as exploration strategy. With this, the agent creates its own dataset in order to compute G_ω.

Proximal Policy Optimisation (PPO) [24]: On-policy gradient approach that limits the size of the policy update by clipping the difference between the current and old policy. Consequently, the training becomes more stable and it prevents the policy from falling off the cliff. Its clipped objective function L^{CLIP} is denoted in (7), where $\hat{A}$ is the advantage. Its final objective function also consists out of a value loss and entropy component to ensure sufficient exploration.

$$\begin{aligned} L^{\text{CLIP}}(\theta) \doteq \hat{\mathbb{E}}\Big[\min\Big(\frac{\pi_\theta(a|s)}{\pi_{\theta_{\text{old}}}}\hat{A}, \\ \text{clip}\Big(\frac{\pi_\theta(a|s)}{\pi_{\theta_{\text{old}}}}, 1-\epsilon, 1+\epsilon\Big)\hat{A}\Big)\Big] \end{aligned} \tag{7}$$

In addition, we have also included its recurrent variant; rPPO. Here, the forward layers have been replaced by LSTM [25] layers.

Multi Dynamics- and Q-Learning (MDQL). Online off-policy algorithm that makes use of shared state representation learning between a set of ensembles to make the learning more robust. It is an extension to DQN, and learns the transition and reward dynamics next to the state-action values. The predictions that MDQL can make are summarised in (8), where $\mathcal{F}$ is the shared feature extractor, Q, $\hat{D}$ and $\hat{R}$ the ensembles that learn the q-value, transition and reward dynamics, respectively. Variable i is a member of the ensemble, and $s_\triangle$ the state difference such that $s' = s + s_\triangle$.

$$\begin{aligned} &Q_{\theta,i}\big(\mathcal{F}_\theta(s)\big) \mapsto q_i \\ &\hat{D}_{\theta,i}\big(\mathcal{F}_\theta(s), a\big) \mapsto s_{\triangle,i} \\ &\hat{R}_\theta\big(\mathcal{F}_\theta(s), a\big) \mapsto \hat{r} \end{aligned} \tag{8}$$

At decision making time, the action is selected based on the aggregation of the Q-members. As aggregation strategy, we take the Confidence Lower Bound [26] over the estimations. Consequently, we penalise actions with high disagreement between the Q-heads, which should be higher in state-action regions with sparse datapoints. Thus, we penalise actions that have high epistemic uncertainty. We also introduce its recurrent counterpart, rMDQL, where the forward layers in $\mathcal{F}$ are replaced by LSTM layers.

VI. Experimental Setup

In this section, we cover the main details and configurations regarding the experimental setup. All experiments have been conducted on a shared compute cluster, from which we requested a compute unit with the following specifications:

- CPU: 1 Intel Xeon Gold 6248 core @ 2.50 GHz
- GPU: 1 NVIDIA Tesla V100 (32GB HBM2 VRAM)

Additionally, we report on the mean and standard deviation over three replications. Learn curves are smoothed with the Savitzky-Golay filter, with its implementation taken from SciPy [27]. The window length is set to 21 and the order of the polynomial fitted to the samples is set to 4.

A. Environment Configuration

Since we are dealing with a non-traditional RL environment (see Section IV) due to the presence of three years of historical data, we have opted to split the data into a train, validation and test set, with a split of 26:5:5. The training set contains the first 26 months, while the remaining months are divided over the validation and test set by alternating between the sets when assigning the next month. The reasoning behind this assignment strategy is to ensure that the validation and test sets contain roughly the same number of spring, summer, fall and winter months. See below for an overview of the three sets:

- **Train set**: 2017, 2018, {January, February} of 2019.
- **Validation set**: {April, June, August, October, December} of 2019.
- **Test set**: {March, May, July, September, November} of 2019.

Furthermore, a training episode takes 1,420 hours (i.e., ∼2 months), irrespective of the action taken by an optimisation scheme. The RL methods are trained for 100 epochs, where each epoch consists of 20 training episodes. Subsequently, each model is trained on 2.8M timesteps. During evaluation, *all* five months are included in an evaluation round, where an episode takes exactly one month. Subsequently, each evaluation round consists of five episodes. The attributes that are not based on historical data are randomised at the start of an episode.

Lastly, the reward function is configured such that the battery SoH-penalty is only applied to the `NETTING` action, with its scalar λ_{SoH} set to 1. The penalty multiplier m is set to 1.05.

B. Hyperparameter Configuration

The hyperparameter configuration of each RL algorithm largely follows the default configuration set by the used implementation. For DQN and (r)PPO, we used the Stable-Baselines3 (Contrib) library [28], and for BCQ we used the author's implementation which can be found at `https://github.com/sfujim/BCQ`. As for MDQL, the majority of the hyperparameter settings have been copied from DQN. The remaining settings, such as the ensemble size and learning rate, have been optimised specifically for MDQL.

A subset of the hyperparameters have been optimised through grid search. The resulting best found values have been summarised in Table I.

TABLE I
THE VALUES FOR EACH HYPERPARAMETER THAT HAS BEEN OPTIMISED. IN CASE A HYPERPARAMETER IS NOT APPLICABLE TO AN ALGORITHM (I.E., GAE FOR DQN), WE INSERT '×'. NOTE THAT THERE ARE NO LAYERS IN THE Q-ENSEMBLE FOR (R)MDQL, RESULTING IN A LINEAR MAPPING FROM THE FEATURE EXTRACTOR TO THE ESTIMATED q-VALUES.

Hyperparameter	**DQN**	**BCQ**	**(r)PPO**	**(r)MDQL**
Network arch.	[256, 256]	[64, 64, 64]	[256, 256]	$\mathcal{F}$: [1024, 1024] Q: — $\hat{D}$: [256, 256] $\hat{R}$: [128, 128]
Learning rate	0.001	0.00001	0.0001	0.0001
Initial ϵ	1.00	1.00	×	0.90
Final ϵ	0.05	0.05	×	0.40
Φ_{BCQ}	×	0.10	×	×
GAE	×	×	0.95	×
Entropy coef.	×	×	0.01	×
Ensemble size	×	×	×	4

VII. Experimental Results

By means of experimentation, we study how all optimisation approaches compare against each other, and discuss an analysis on the decision making process of MDQL.

A. Benchmarks

In this experiment, we evaluate all algorithms against each other. First, we show the smoothed learning curves on the validation environment in Figure 1. It shows how MDQL, and to a lesser extent rMDQL, outperform all benchmarks, with the exception of MILP. MDQL has an asymptotic learn curve, while rMDQL slightly drops off after its initial peak at 500k timesteps. Noteworthy is the curve of DQN, which initially shows a similar trend as MDQL. However, it then drops off significantly to the level of the 'Idle'-benchmark. This is the result of diverging q-values due to bootstrapping. It is caused by the fact that the environment returns a negative reward for almost all actions.

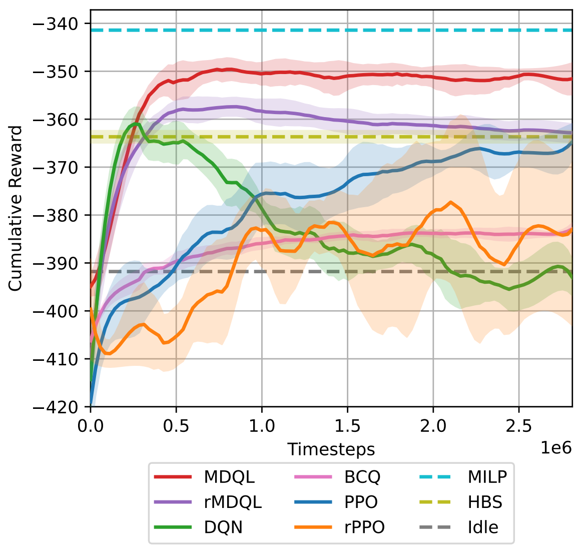

Fig. 1. Smoothed learn curves on the validation environment. Algorithms without a learn curve are denoted with the dashed lines. Note that the parameters of the HBS are optimised on both the training and validation environment. As the figure shows, MDQL performs the best out of all benchmarks, barring MILP with perfect foresight (which serves to quantify an upper bound). Somewhat surprisingly, adding recurrent layers to the feature extractor results in a measurable deterioration of the overall performance.

Moreover, the difference between DQN and BCQ is noteworthy. It shows that the offline aspect of the environment plays a minimal role in obtaining a high cumulative reward. Instead, constraining the updates to the data distribution resulted in a conservative policy, since BCQ only barely outperforms the lower bound.

Lastly, PPO shows a stable learning curve and obtains a performance similar to HBS. Notably, PPO seems to not have settled yet, and more training time might result in approaching or exceeding MDQL. Its recurrent counterpart proves to be rather unstable, with the widest error out of all algorithms. This, along with the performance of rMDQL, demonstrates that recurrent layers seem to be more sensitive

TABLE II
METRICS OF EACH METHOD ON THE TEST ENVIRONMENT. FOR EACH METRIC, WE REPORT ON THE MEAN AND (IF APPLICABLE) STANDARD DEVIATION, WHICH ARE TOP AND BOTTOM PER METRIC, RESPECTIVELY. IN ADDITION, WE HIGHLIGHT THE OVERALL BEST METRICS IN **BOLD**, WHILE WE ALSO UNDERLINE THE BEST METRICS EXCLUDING MILP. LASTLY, FOR NETTING, WE ONLY CONSIDER THE REPLICATIONS THAT HAVE NETTED SOME POWER.

Metric	**Idle**	**MILP**	**HBS**	**DQN**	**BCQ**	**PPO**	**rPPO**	**MDQL**	**rMDQL**
PV util. (%)	0.00	64.00	52.63	72.32	50.27	74.73	75.69	**76.54**	74.01
			4.40	13.96	1.74	9.44	12.45	2.67	0.03
Montly costs	89.07	**79.87**	83.13	82.35	86.49	82.99	82.72	80.78	81.92
			0.29	0.52	0.09	0.53	1.17	0.30	0.57
Cum. reward	-445.34	**-400.69**	-417.89	-414.35	-436.94	-418.24	-421.55	-405.76	-413.84
			1.29	1.98	0.61	1.69	7.93	1.06	3.64
Consumption									
No. MWh	8.73	8.38	8.45	8.37	8.48	8.35	8.35	**8.35**	8.36
			0.02	0.07	0.01	0.04	0.05	0.01	0.01
Costs	445.34	**399.59**	415.65	411.77	433.51	414.96	413.62	403.97	409.75
			0.01	2.56	0.31	2.66	5.84	1.48	2.91
Mean tariff	0.051	**0.048**	0.049	0.049	0.051	0.050	0.050	**0.048**	0.049
			0.000	0.000	0.000	0.000	0.001	0.000	0.000
Netting									
No. kWh	0.00	3.96	0.00	0.91	**16.66**	0.42	0.00	0.86	3.01
			0.00	1.28	6.13	0.59	0.00	0.67	1.40
Profits	0.00	0.24	0.00	0.04	**1.06**	0.02	0.00	0.05	0.16
			0.00	0.05	0.39	0.03	0.00	0.04	0.08
Mean tariff	—	0.061	—	0.050	**0.064**	0.042	—	0.057	0.053
				—	0.001	—		0.000	0.002

in terms of hyperparameter settings. Moreover, the benefit also seems to be underwhelming, which can be attributed to the fact that $\mathcal{S}$ already provides weather forecasts for the near future.

Next, we take the best weights of each algorithm and evaluate them on the test environment, from which we denote a set of metrics in Table II. Noteworthy is that no method makes extensive use of the `NETTING` action. The action is selected in rare occassions, mainly when the grid tariff is relatively high. It demonstrates that only with high tariffs it is possible to overcome the battery SoH-penalty. This behaviour indicates that it is not cost-effective to use the battery, since the operating costs outweigh the return. To solve this, cheaper and/or more durable batteries as well as higher grid tariffs seem to be necessary to reach and surpass the crossover point in order to make `NETTING` worthwhile.

The available power is instead consumed by the heatpump. Compared to the 'Idle'-baseline, all other methods consume less power from the grid, which can accumulate up to 0.5 MWh's over the five test months. MDQL and PPO consume the least amount of MWh, closely followed by MILP despite it obtaining the highest cumulative reward. Overall, the methods are able to save up to nine euros each month.

Interestingly, all methods have a PV-utilisation below 80%. This can be explained by the fact that the battery is only able to perform one action at a time, so achieving a PV-utilisation of 100% *and* having a high cumulative reward is extremely difficult, if not impossible. It would mean that in order to achieve this, the battery must charge continuously during daytime, and is only able to discharge during nighttime. This in itself is not cost-effective, since the general trend of grid tariffs have shown to be higher during daytime than nighttime. So instead, it is optimal to sometimes waste PV-power. For instance, MILP has one of the lowest PV-utilisations (64%) and a higher grid consumption than MDQL, and makes up for this by exploiting the grid tariffs to a greater extent.

B. MDQL Decision Making Analysis

In order to analyse the behaviour and reasoning of the agent, we have plotted one week from the test environment in Figure 2. During this week, the agent mainly charges the battery via the PV panels. As the figure shows, the majority of the available PV power has been covered by the agent. Then, this power is consumed by the heatpump in the hours where there is little to no PV radiation. In addition, the agent sometimes opts for a quick charge from the grid at midnight, possibly since the grid tariffs are often relatively low during that point in time. Unfortunately, these drops in tariff are not fully exploited. For instance, the agent does not opt to charge the battery during the low tariff at the end of the 6th of July, despite the battery being nearly depleted. Another example would be at the start of the 2nd of July, where the drop should have been used to charge the battery.

VIII. DISCUSSION

IN this work, we have benchmarked a set of optimisation algorithms on the battery dispatch setting, with a focus on reinforcement learning (RL) approaches. In addition, we propose Multi Dynamics- and Q-Learning (MDQL), which is a novel extension to the Deep Q-Network algorithm by sharing state representations between q-, dynamics- and reward-approximators for weight regularisation.

Based upon the experiments, we demonstrate that the RL approaches fall short of Mixed-Integer Linear Programming (MILP) with perfect foresight, which quantifies an upper

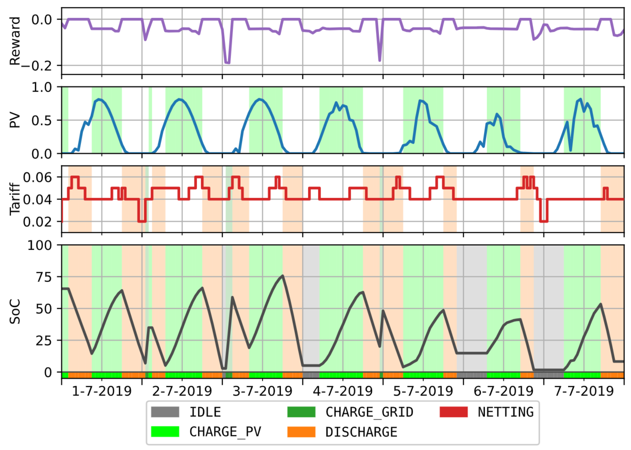

Fig. 2. MDQL decision making over one week in July. It consists out of four plots, from top to bottom: the reward signal, PV radiation in kWh, grid tariff in €/kWh and progression of the battery SoC in percentages. Moreover, the background color denotes the action that has been selected at that point in time. As the figure shows, the agent tries to maximise its PV-utilisation, and deploy this power during nighttime.

bound on the performance[4]. Among the RL algorithms, MDQL obtains and settles to the highest cumulative reward. Noteworthy is how recurrency introduces instability and/or an overall worse performance. This instability is also present in DQN, where it falls off the cliff at 500k timesteps, after which it settles to the performance of the 'Idle'-baseline. Overall, PPO, MDQL and rMDQL obtain asymptotic learn curves, and are able to match or exceed the expert system.

A decision making analysis of MDQL shows that the strategy consists of maximising the PV utilisation, and deploying it to the household outside sun-hours. In addition, the battery is often charged from the grid at midnight since the power is relatively cheap at that point in time. However, the analysis also shows that there are still gains to be made by exploiting the spikes more effectively, as is demonstrated by MILP. In the current problem setting, this can be at least 1 euro every month, but with higher grid tariffs the potential savings will become more significant.

Moreover, the lack of `NETTING` shows how the battery SoH penalty is too severe to overcome. It indicates that the current setup is not cost-effective in order to use the `NETTING` action. Cheaper batteries and/or higher grid tariffs might be required in order to overcome the cost of purchase.

One of the main future work suggestions is regarding the MILP baseline. Currently, MILP has access to the perfect information, resulting in an unfair comparison against the other approaches. In addition, a set of constraints have been imposed on MILP due to its computational properties. This does raise the question as to how its performance would be with less restrictive constraints and making its decisions based on forecasts or historical data.

Other possible future work directions could lie in improving MDQL by utilising the learned transition- and reward-dynamics for decision-time planning. Subsequently, MDQL will become model-based, and we believe that planning at testing time should result in performance improvements. Another important aspect to investigate is the computational costs of the algorithms from the differing domains in case of real-world deployment. In general, RL poses to be a promising approach to battery dispatch optimisation, due to its low deployment costs compared to MILP and overall performance.

Lastly, more accurate/realistic battery dynamics and state of health approximations might result in a more nuanced perspective of discharging to the power grid (i.e., `NETTING`) or to the household, as it has been shown that it is not cost effective in the problem setting to take the battery lifespan into consideration due to its cost of purchase.

[4]Note: this is not the absolute upper bound, due to a set of constraints (time budget, limiting the horizon) that have been imposed on MILP.

ACKNOWLEDGEMENTS

This research has been carried out in collaboration with Shell Global Solutions International B.V. The data used in this work is owned by Shell. The authors would like to acknowledge the compute resources provided by Shell.

REFERENCES

[1] U. Zafar, S. Bayhan, and A. Sanfilippo, "Home energy management system concepts, configurations, and technologies for the smart grid," *IEEE access*, vol. 8, pp. 119 271–119 286, 2020.
[2] J. R. Vázquez-Canteli and Z. Nagy, "Reinforcement learning for demand response: A review of algorithms and modeling techniques," *Applied energy*, vol. 235, pp. 1072–1089, 2019.
[3] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.
[4] M. D. de Souza Dutra, M. F. Anjos, and S. Le Digabel, "A realistic energy optimization model for smart-home appliances," *International Journal of Energy Research*, vol. 43, no. 8, pp. 3237–3262, 2019.
[5] T. Yu, D. S. Kim, and S.-Y. Son, "Optimization of scheduling for home appliances in conjunction with renewable and energy storage resources," *Int. J. Smart Home*, vol. 7, no. 4, pp. 261–272, 2013.
[6] M. Hu and F. Xiao, "Price-responsive model-based optimal demand response control of inverter air conditioners using genetic algorithm," *Applied energy*, vol. 219, pp. 151–164, 2018.
[7] H. M. Lugo-Cordero, A. Fuentes-Rivera, R. K. Guha, and E. I. Ortiz-Rivera, "Particle swarm optimization for load balancing in green smart homes," in *2011 IEEE Congress of Evolutionary Computation (CEC)*. IEEE, 2011, pp. 715–720.
[8] E. Matallanas, M. Castillo-Cagigal, A. Gutiérrez, F. Monasterio-Huelin, E. Caamaño-Martín, D. Masa, and J. Jiménez-Leube, "Neural network controller for active demand-side management with pv energy in the residential sector," *Applied Energy*, vol. 91, no. 1, pp. 90–97, 2012.
[9] B. Yuce, Y. Rezgui, and M. Mourshed, "Ann–ga smart appliance scheduling for optimised energy management in the domestic sector," *Energy and Buildings*, vol. 111, pp. 311–325, 2016.
[10] M. S. Ahmed, A. Mohamed, R. Z. Homod, and H. Shareef, "Hybrid lsa-ann based home energy management scheduling controller for residential demand response strategy," *Energies*, vol. 9, no. 9, p. 716, 2016.
[11] C. A. Hernandez, R. Romero, and D. Giral, "Optimization of the use of residential lighting with neural network," in *2010 International Conference on Computational Intelligence and Software Engineering*. IEEE, 2010, pp. 1–5.
[12] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.
[13] Y. Chen, L. K. Norford, H. W. Samuelson, and A. Malkawi, "Optimal control of hvac and window systems for natural ventilation through reinforcement learning," *Energy and Buildings*, vol. 169, pp. 195–205, 2018.
[14] Y. Wang, K. Velswamy, and B. Huang, "A long-short term memory recurrent neural network based reinforcement learning controller for office heating ventilation and air conditioning systems," *Processes*, vol. 5, no. 3, p. 46, 2017.

[15] S. Bahrami, V. W. Wong, and J. Huang, “An online learning algorithm for demand response in smart grid,” *IEEE Transactions on Smart Grid*, vol. 9, no. 5, pp. 4712–4725, 2017.
[16] B.-G. Kim, Y. Zhang, M. Van Der Schaar, and J.-W. Lee, “Dynamic pricing and energy consumption scheduling with reinforcement learning,” *IEEE Transactions on smart grid*, vol. 7, no. 5, pp. 2187–2198, 2015.
[17] B. V. Mbuwir, F. Ruelens, F. Spiessens, and G. Deconinck, “Battery energy management in a microgrid using batch reinforcement learning,” *Energies*, vol. 10, no. 11, p. 1846, 2017.
[18] Z. Tan, X. Zhang, B. Xie, D. Wang, B. Liu, and T. Yu, “Fast learning optimiser for real-time optimal energy management of a grid-connected microgrid,” *IET Generation, Transmission & Distribution*, vol. 12, no. 12, pp. 2977–2987, 2018.
[19] M. L. Puterman, *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.
[20] J. Muñoz-Sabater, E. Dutra, A. Agustí-Panareda, C. Albergel, G. Arduini, G. Balsamo, S. Boussetta, M. Choulga, S. Harrigan, H. Hersbach *et al.*, “Era5-land: A state-of-the-art global reanalysis dataset for land applications,” *Earth system science data*, vol. 13, no. 9, pp. 4349–4383, 2021.
[21] J. Wang, P. Liu, J. Hicks-Garner, E. Sherman, S. Soukiazian, M. Verbrugge, H. Tataria, J. Musser, and P. Finamore, “Cycle-life model for graphite-lifepo4 cells,” *Journal of power sources*, vol. 196, no. 8, pp. 3942–3948, 2011.
[22] S. Fujimoto, D. Meger, and D. Precup, “Off-policy deep reinforcement learning without exploration,” in *International conference on machine learning*. PMLR, 2019, pp. 2052–2062.
[23] S. Fujimoto, E. Conti, M. Ghavamzadeh, and J. Pineau, “Benchmarking batch deep reinforcement learning algorithms,” *arXiv preprint arXiv:1910.01708*, 2019.
[24] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal policy optimization algorithms,” *arXiv preprint arXiv:1707.06347*, 2017.
[25] H. Sak, A. Senior, and F. Beaufays, “Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition,” *arXiv preprint arXiv:1402.1128*, 2014.
[26] J. Smit, C. T. Ponnambalam, M. T. Spaan, and F. A. Oliehoek, “Pebl: Pessimistic ensembles for offline deep reinforcement learning,” in *Robust and Reliable Autonomy in the Wild Workshop at the 30th International Joint Conference of Artificial Intelligence*, 2021.
[27] P. Virtanen, R. Gommers, T. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright *et al.*, “Fundamental algorithms for scientific computing in python and scipy 1.0 contributors. scipy 1.0,” *Nat. Methods*, vol. 17, pp. 261–272, 2020.
[28] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, “Stable-baselines3: Reliable reinforcement learning implementations,” *The Journal of Machine Learning Research*, vol. 22, no. 1, pp. 12 348–12 355, 2021.

Jerry Schonenberg received the B.Sc. and M.Sc. Computer Science degrees from Leiden University, Leiden, The Netherlands, in 2020 and 2023, respectively. During this period, he has been active as teaching assistant for various computer science courses given at the university. His research interests include reinforcement learning, computer vision and natural language processing.

Kapil Mathur was born in Udaipur in the Rajasthan, India, on March 23, 1983. He graduated from the Gyan Vihar College of Engineering and studied at the University of Rajasthan. Post Graduation from Centre for development of Advanced Computing. His employment experience included Senior Technical Officer at C-DAC R&D, Pune, India and Principle IoT & Edge Computing SME in Shell Technology Centre Bangalore. His specialization in the fields of High Performance Computing, with the focus on weather forecasting, seismic data processing and reservoir modelling, along with Reinforcement learning, IoT & Edge Computing for energy optimization.

Carlos Ros Perez was born in Valencia, Spain. He graduated with a Bachelor’s degree in Physics from the University of Valencia and an MSc in Applied Mathematics at Imperial College London. His professional experience includes a research internship in computational biology at I2SysBio. Currently, he is working as an AI researcher for power markets at Shell, where he contributes to projects focused on power dispatch and renewable energies.

Detlef Hohl holds a Master’s degree in chemistry from the Technical University of Munich and a Ph.D. in theoretical physics from the Technical University of Aachen. Before joining Shell in 1997, he was senior staff member at Forschungszentrum Jülich, Germany’s largest National Laboratory to this day. He spent research assignments at SISSA Trieste, Rice University, Stanford University, the University of Illinois (NCSA) and NIST. Detlef executed and managed many projects in geophysics and reservoir engineering in Shell. The focus of his entire career has been the computer simulation of physical phenomena, often using the fastest and most expensive high-performance computers available at the time. He is Shell’s Chief Scientist for Computational and Data Science and also holds an adjunct professor of Computer Science appointment at Rice University and University of Houston.

Aske Plaat is professor in artificial intelligence at Leiden University, Scientific Director of the Institute of Artificial Intelligence and Computer Science (LIACS), and heads the Sustainable Energy Learning Lab and the Reinforcement Learning Group. He is the author of two textbooks on Deep Reinforcement Learning. His research interests are in reinforcement learning and self-learning systems. He has won the Novag Award for the best computer chess publication. He is co-author of an ERC Advanced grant on combinatorial algorithms for high energy physics. Previously he has worked at the Vrije Universiteit Amsterdam, Tilburg University, the University of Alberta, and the Massachusetts Institute of Technology.

Thomas Moerland obtained a PhD from TU Delft, focusing on model-based reinforcement learning. He currently works as an assistant professor at the Leiden Institute of Advanced Computer Science.

Christian Michler is a Principal AI Researcher with Shell Global Solutions International B.V. in The Netherlands. He graduated from the Technical University of Clausthal and did post-graduate studies at von Karman Institute for Fluid Dynamics. He received his PhD from the Technical University of Delft, and did post-doctoral research at The University of Texas at Austin and the University of Oxford. Dr. Michler held a Lecturer Position / Assistant Professorship at King’s College London before joining Shell Global Solutions International B.V.