Received: 10 July 2020 | Revised: 12 September 2020 | Accepted: 15 September 2020

DOI: 10.1002/er.6023

RESEARCH ARTICLE

INTERNATIONAL JOURNAL OF ENERGY RESEARCH WILEY

Optimal peak shifting of a domestic load connected to utility grid using storage battery based on deep Q-learning network

Mostafa Al-Gablawy

Power Engineering and Automatic Control Department, Pyramids Higher Institute for Engineering and Technology, Giza, Egypt

Correspondence
Mostafa Al-Gablawy, Power Engineering and Automatic Control Department, Pyramids Higher Institute for Engineering and Technology, Egypt.
Email: mostafagabalawy@gmail.com

Summary

Peak periods are a result of consumers generally using electricity at similar times and periods as each other, for example, turning lights on when returning home from work, or the widespread use of air conditioners during the summer. Without peak shifting, the grid's system operators are forced to use peaked plants to provide additional energy. This operation is incredibly expensive and harmful to the environment due to its high levels of carbon emissions. Battery storage system (BSS) has been proposed to allow purchasing the energy during off-peak periods for later use, with the primary objective of realizing peak shifting occurred. Multi-objective optimization with the reinforcement learning technique has been utilized in order to achieve the primary objective, reduce energy consumption, and minimize the consumers' utility bills. The results revealed that the reduction in energy consumption was more than 20%, the consumers' energy bills were minimized, as well as realizing perfect peak shifting. In addition, the strategy attempted to overcharge the battery with about 7% of the time, and promising methods to address this has been proposed as a direction for future research.

KEYWORDS

battery storage system (BSS), deep Q-learning network, deep reinforcement learning, optimal peak shifting, peak shifting

1 | INTRODUCTION

A significant problem in the generation and consumption of energy is that of peak demand.[1] Although energy usage varies from household-to-household, season-to-season, and many other factors, such as location, there are still numerous trends in energy usage, which could be predicted.[2] For instance, when people return home after work, they may turn on the television, start cooking, or may turn on the air conditioner if the temperature is far from comfortable before sleeping.[3] These trends result in times of significantly higher energy demand, known as peak periods, and could be incredibly difficult for the grid to manage.[4] At times of high demand, peak power plants are generally needed to be used to provide the additional required energy.[5] It is a particular case when moving towards the intermittent decentralized network at times when the forecasted demand for energy is significantly lower than the actual demand. These peak power plants are harmful to the environment as they are generally generating energy through inefficient combustion of fossil fuels, and are also incredibly expensive to operate.[6] These higher costs are covered by the consumer, resulting

The data that support the findings of this study are available from the corresponding author upon reasonable request.

in much higher than necessary utility bills. In addition, peak power plants cannot be turned on instantly and generally might take at least an hour to become functional, adding yet another layer of uncertainty and potential for power outages at times of high demand.[7]

Many load shifting studies in power systems have been conducted in terms of the optimum storage configuration. Shi et al[8] proposed a joint optimization technique using the BSS simultaneously for the sake of frequency regulation and peak shaving that captures battery regression, operating constraints, and customer load degradation. Their study revealed that the user's electricity bill could be minimized by up to 12%. Setlhaolo et al[9] proposed a nonlinear programming mathematical method (NPMM) with the BSS in order to optimize the household appliances regression under the time of utilizing electricity tariffs. Their study demonstrated the ability of that algorithm in load shifting by means of optimizing appliances degradation and battery, valley filling, charge saving, and peak shaving. Hashim et al[10] presented a mixed-integer linear program (MILP) in order to build a combined biomass solar city that involved the BSS and load shifting. Their study revealed that the loads were optimally shifted while the BSS was employed to reduce the capacities of the operating units. Barzin et al[11] presented a control scheme to achieve optimum load shifting for building sectors with the incorporation of the BSS technology. Their approach achieved cost savings of more than 60%. Uddin et al[12] discussed the peak load shifting policies with the incorporation of the BSS and electric vehicles (EVs). Their study proved that the BSS could be an appropriate tool for shifting peak loads. Rozali et al[13] presented a power pinch study to estimate peak load shifting for optimum storage sizing in addition to optimize the subscriber's bills. Taylor et al[14] proposed the BSS operation based on a stochastic optimization for the sake of peak shaving at 11 kV distribution system. Ananda-Rao et al[15] proposed different types of the BSS for peak shaving with controlling in charging/discharging operations as well as extended the lifetime of the BS schemes. Bao et al[16] proposed a peak demand with a BSS by utilizing the mixture of regression methods for load profile forecasting and dynamic programming (DP) for optimal action selection, where action referred to buy and store energy. Qin et al[17] used an online modified greedy algorithm. Their study was addressed as computational complexity suffered by DP methods, but with the additional problem of requiring bounds on the pricing structure and not being able to constrain the storage size of the BSS. Kim and Lim[18] presented the problem of BSS utilizing a reinforcement learning approach. Their approach could learn a policy independent of the price distribution and therefore could perform well even in an environment with a nonstationary price profile.

Deep Q-learning has been applied for optimum operations of the BSS in many studies. Bui et al[19] proposed a dual deep Q-learning approach for dealing with the distributed BSS operation in a microgrid system. Shen et al[20] provided a deep learning approach, focused on the calculation of the voltage, current, and charging power of the BSS operation over a part-load period, using a deep-repressive neural network (DRNN). Ren et al[21] presented the deep-learning framework combined with the DRNN for retains useful life (RUL) forecast of a lithium-ion (Li-ion) battery. Meng et al[22] presented an estimation for the battery state of health (SOH) by utilizing a deep learning approach that could improve the efficiency of the data-driven SOH assessment through well-designed incorporation of the feeble learners. Shen et al[23] proposed a deep learning-based approach with the integration of ensemble learning for estimating the capacity of the Li-ion battery with a view to achieving satisfactory accuracy. Bhowmik et al[24] proposed the generative method of the machine learning framework for the monitoring and enhancement of reliability in the predictive design of battery interfaces. Yang et al[25] proposed an approach to the charge/discharge and the purchase schedule for the BSS based on deep reinforcement learning (DRL). Chemali et al[26] proposed a new technique utilizing the DRNN in order to estimate the state of charge (SOC) of the BSS interface, where the battery measurements were charted directly into SOC. Liu et al[27] reviewed the machine learning approaches for the design of rechargeable batteries materials, including property forecasting for solid and liquid electrolytes and electrode materials, with discovering the forecast of components and structure of new rechargeable battery materials. Sedighizadeh et al have applied a stochastic optimization algorithm for scheduling a short term in a microgrid that consists a storage battery. Many power sources are considered in this microgrid such as; wind turbine, PV system, storage battery, microturbine, and fuel cell. A cost function has been designed considering the uncertainty of the wind speed and solar irradiation. A hybrid algorithm from modified particle swarm with the Differential Evolutionary is applied to deterministically solve the objective function. Here, the authors try to make a scheduling for a short term for this microgrid and depending on the storage energy systems such as fuel cell and batteries.[28] While, in Ref. 29 the authors apply reconfiguration for IEEE 33 bus system feeders as well as the scheduling for the generators using GAME software considering all operational constraints. Mainly, the cost function is designed to consider the active and the reactive power from the utility grid and the switching cost to obtain the best distribution feeders configuration.

This paper outlines a solution method to the peak demand problem utilizing the BSS interface, with the

goal of learning an optimal policy of buying and storing energy in an incredibly dynamic and complex non-stationary environment via the usage of reinforcement learning. The BSS has been chosen due to the fact it is becoming more feasible in recent times due to lowering costs, as well as it is easy to install, repair, replace, and is being used by many companies to-date, verifying its effectiveness. However, a simple battery is easy to model in detail. Based on the data made available by informatics, the end-user would be various households situated in Japan. This is an important stipulation, as the size of batteries used by households would be significantly different from that used at large facilities, and of course, as different countries exhibit varying energy protocols, as discussed previously. It should be noted that the use of batteries introduces some additional complications. First, capacity degradation, a phenomenon whereby the maximum capacity, or upper limit of stored energy, depletes over time. Second, self-discharge is another degrading phenomenon of batteries, which refers to the stored energy being automatically discharged over time. These issues would affect the optimal policy of buying and storing energy, but they would be ignored due to their relatively small effect. The objective is not only to provide the exact optimal policy for a specific BSS for a specific household but also to verify the effectiveness of reinforcement learning to the peak shifting solution, where the choice of country, end-user, and BSS simply are conveniences to aid in testing and evaluation. The rest of the paper is organized as follows: Section 2 illustrates the system description; the proposed algorithm of deep Q-learning method is discussed and illustrated in detail in Section 3; the relation to the peak shifting domain is illustrated in Section 4, the detailed results and simulations are described in Section 5, and finally the conclusions are illustrated in Section 6.

2 | SYSTEM DESCRIPTION

A comprehensive understanding of this data would allow for a more informed decision on which reinforcement learning algorithms are most applicable to this domain. The datasets for this task consist of what will be referred to as observational and household data, each consisting of minute-by-minute data points over a period of 6 months. The observational data comprises three different datasets; the prices of electrical energy from the grid, weather data, and demand data, which is the technical term for the limit of available energy from the grid. As demand could also refer to how much energy is required by the household, these terms would be used interchangeably throughout, but the meaning would always be clear from the context. The price and demand datasets consist of multiple different plans, that is, different pricing and demand profiles, but during the model learning phase, only one plan for each would need to be chosen. Figure 1 presents 3 days of pricing data using four different pricing structures; smart life plan, night-8, night-12, and random. Random is artificially created data, while the others are referring to actual pricing plans offered by the electric power company. Night-8 and night-12 are plans for customers, who use more electricity at nighttime, while the smart life plan is for customers moving towards smart homes, who would, in general, be using electricity a lot throughout both day and night.[28] An algorithm should be able to train a model irrespective of which pricing plan is chosen, assuming there is some sort of structure to be learned. For this reason, random would not be used as it simply introduces unnecessary noise into the learning process. During the testing stage, one of the remaining plans would be chosen at random. Figure 2 gives four different meteorological datasets that could be used to predict energy generation via photovoltaic. The model training process should be flexible

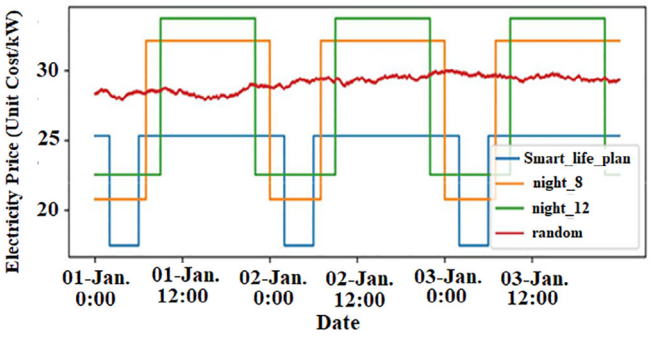

FIGURE 1 Pricing data for the first 3 days of Jan [Colour figure can be viewed at wileyonlinelibrary.com]

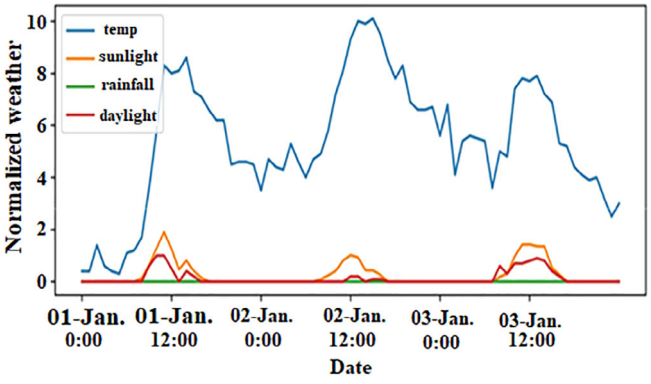

FIGURE 2 Normalized weather data giving the temperature, sunlight, rainfall, and daylight for the first 3 days of Jan [Colour figure can be viewed at wileyonlinelibrary.com]

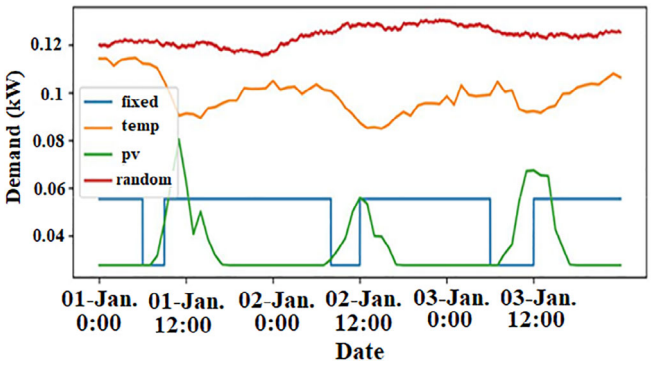

FIGURE 3 Four different artificially created demand plans over the first 3 days of Jan [Colour figure can be viewed at wileyonlinelibrary.com]

enough to allow for any combination of these datasets to be used, noting that it might be possible to accurately forecast energy generated without the need for the rainfall data, for example. Although the data presented has been normalized, this does not affect the learning process as all data would be standardized before being used. In addition, this data does not necessarily represent the actual weather at the household, but rather at the closest meteorological site to the household. The difference in location would introduce some noise into the forecasting. Figure 3 presents four different demand plans over the first 3 days of January. These data show that demand refers to the maximum amount of energy available to be purchased at any time. It is necessary that the model is able to forecast the chosen plan well to aid in peak shifting. A model that successfully learns to peak shift would never try to purchase more energy than given by this graph. All data has been artificially created by informatics as public data on-demand profiles that are not yet available in Japan. It should be noted that the data has been slightly modified from the raw values.

All values were divided by 1000 to convert from W to kW, and then again by 60 to convert from kW to kWh, thereby giving the actual energy available for each single-minute time step. Due to a numerical error in the provided data, where the curator originally attempted to convert from W to Wh but incorrectly multiplied by 60 rather than dividing, this data has then been divided again by 60 to achieve the correct values.

In general, informatics has prepared datasets for four different households in Japan. The datasets could be known as non-intrusive load monitoring data (NILM). When monitoring the energy consumption of a household, the data is simply a single number stating the overall energy consumption in kWh for all appliances and devices. However, NILM is an energy disaggregation method that attempts to provide consumption data of individual appliances to provide a clearer picture of exactly which devices are consuming how much energy at different times. The NILM data has been disaggregated using the software being developed at informatics and should provide an additional useful signal to the agent when forecasting data. For example, being able to exclusively monitor the energy used by the air conditioner could allow the agent to learn how a household responds to varying temperature levels. More clearly, one household may choose to use the air conditioner when the temperature, given by the weather data, is greater than 30°C, whereas another may never use their air conditioner. Without the disaggregated NILM data, the agent would find it much more difficult to learn the energy profiles of these two different households. Each household has an associated ID key, but for the sake of brevity, would be referred to as A, B, C, and D hereon. Table 1 provides a mapping between the original IDs and alphabet keys for future reference.

TABLE 1 Mapping of original household IDs to a simpler alphabet referencing structure

Building style	ID key
A	0002_9100000042
B	0002_9100000112
C	0002_9100000113
D	0002_9100000114

Before using the data to train models, it is important to get a better understanding of its structure. On the first inspection, it is evident that over the 6-month period, there are many NaN values present for some of the households. Table 2 illustrates the NaN counts for each NILM entry. NaN values refer to missing appliances, in the case of the dishwasher for household D, or times when the household temporarily removes the logging device from that device, preventing the NILM data from being measured. It is important to note that all NaN data have been replaced with the mean of that column for that specific household. This is just one choice out of many, such as filling with zeros or interpolating values. Zero-filling was rejected as a result of too much information loss. For the columns with a large number of missing data such as the 5882 counts for household B and the 99 079 counts for the TV column for household C, it is observed that these missing values are all situated in a single period of time, that is, values before and after this period are not NaN. This would make interpolation incredibly difficult, especially considering the seasonal nature of this data and so the interpolation filling-method is also rejected in favor of mean-filling. It should still be noted that mean-filling ignores the variation in data due

TABLE 2 NAN counts for each NILM entry

Field/Household	A	B	C	D
Data	0	0	0	0
Air conditioner	2	5882	1	2
Dish-washer	1	5882	1	260 640
Ecocute	2	5883	1	3
Ih	2	5882	1	3
Main	28	5882	1	125
Microwave	1	10	1	1
Photovoltaic	28	5882	31	125
Refrigerator	10	10	1	2
Cooker	1	9	1	34
TV	3	2	99 079	8
Washer	16	1	1	13
Total	94	35 235	99 119	260 956

to seasonal effects but should result in minimal information loss compared to the other two considered methods. All NILM data have been divided by 60 000 to convert from W to kWh and then again by 60 to correct for a similar mistake as in the demand data explained previously. The main entry refers to the total energy usage, which is the sum of all measured appliances (apart from photovoltaic) plus the consumption of unmetered devices.

Photovoltaic then refers to the energy consumed by the solar panels, which is negative as it is a source of generation rather than consumption. Note that in some cases the photovoltaic value is positive, indicating energy consumption. This is since the controller attached to the solar panel cells could sometimes consume more energy than has been generated. Figures 4 to 7 illustrate the average consumption and generation for each household per day over the 6-month period. Through this data, by observing the peaks and the general trend of the curves, it is immediately clear that the energy profiles of each household are significantly different. As the purpose of this task is to verify that a model for peak shifting is possible in this domain, rather than generating a perfect model for a specific set of data, it has been decided that due to this observation and the large number of missing data for households B, C, and D, that only household A would be used. Another reason for this decision is that given more data, it would be possible to cluster households into a few different energy profiles. Then, once it has been verified that reinforcement learning is a suitable solution method for peak shifting in this domain, it could then be applied separately to each cluster, creating a separate model for each of them. With the minimal data

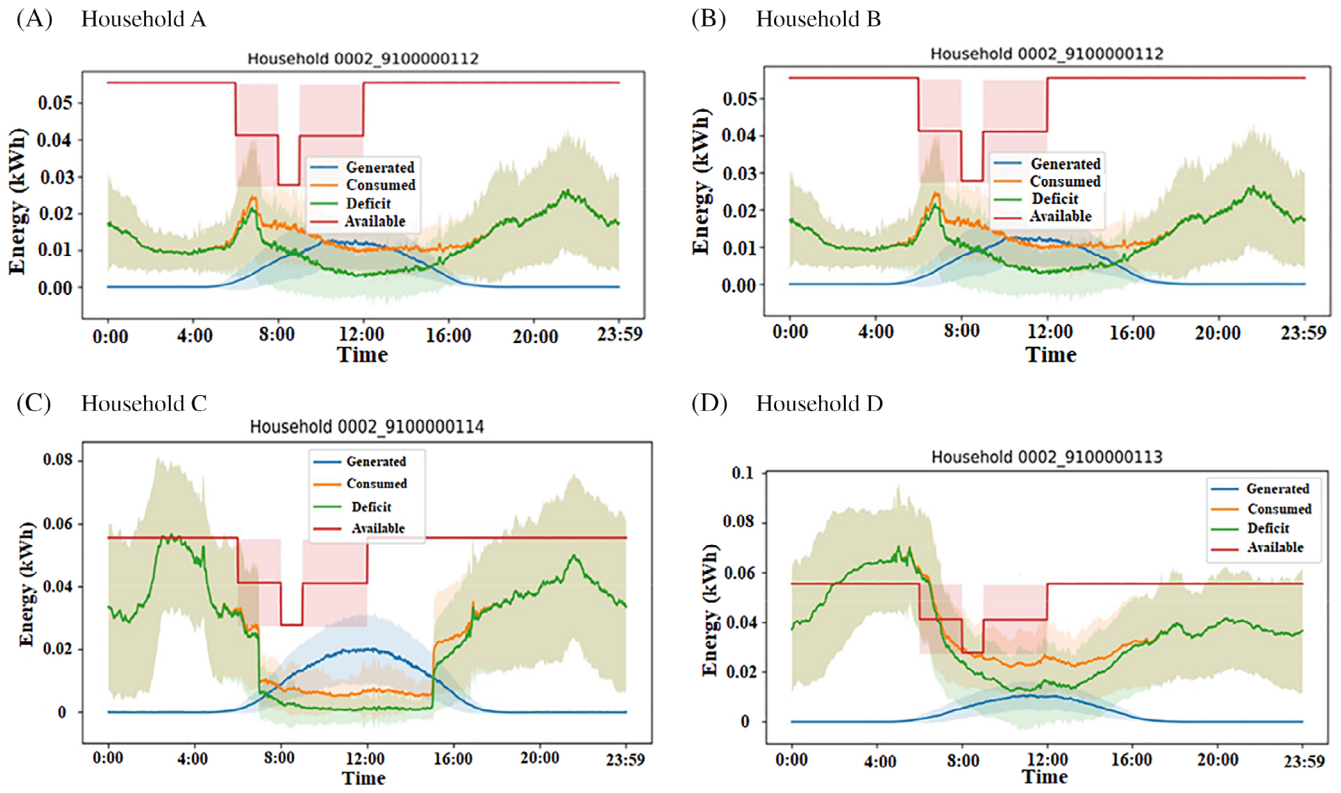

FIGURE 4 Daily averaged generation and consumption data for household A over the 6-month period of available data [Colour figure can be viewed at wileyonlinelibrary.com]

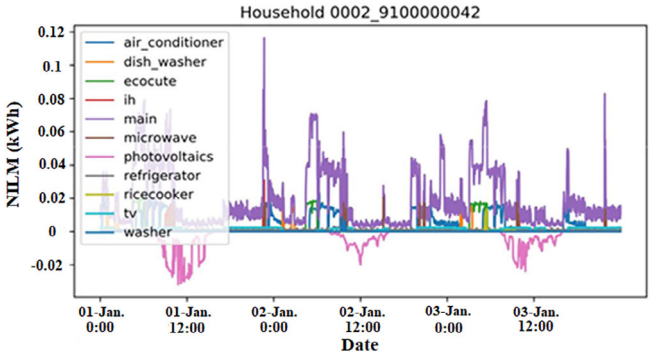

FIGURE 5 NILM data for household A over a 3-day period at the start of Jan [Colour figure can be viewed at wileyonlinelibrary.com]

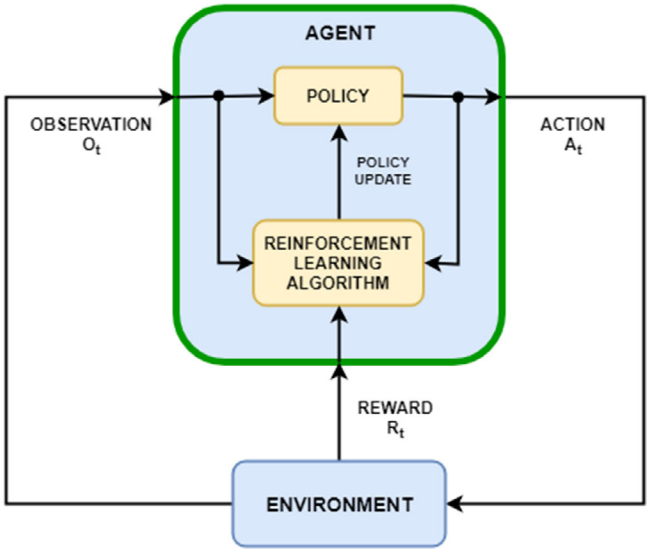

FIGURE 6 Interaction between the agent and environment in an MDP [Colour figure can be viewed at wileyonlinelibrary.com]

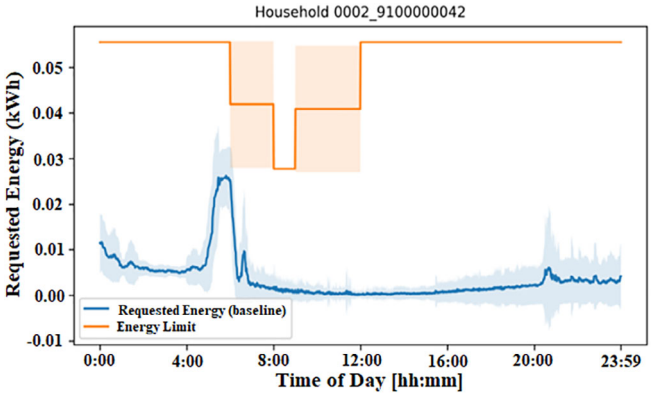

FIGURE 7 Daily-averaged requested an energy profile for the baseline model on household A [Colour figure can be viewed at wileyonlinelibrary.com]

available currently, we would treat household A as the only available dataset of a single cluster, but all methods and code hereafter would be created with the cluster method in mind. Figure 5 shows the disaggregated NILM data for the chosen household A over the first 3 days of January. Here, it is possible to understand better the relative scales of each appliance's energy consumption and the generation owed to the solar panels. As seen from Figure 5, the energy generation from the PV panels occurs during the middle of the day, seen by the larger negative values around noon for each of the 3 days presented.

3 | PROPOSED ALGORITHM

As mentioned previously, the aim will be to verify that reinforcement learning is able to learn a general policy that achieves perfect peak shifting while reducing energy consumption and lowering utility bills without the need to impose various constraints on the problem. More concretely, the method should work irrespective of what price and demand plans are chosen, or what the battery capacity is, or even where the end-user is located. Although the optimal model would vary depending on the values chosen for these variables, the solution method should remain general and unchanged. Markov decision processes (MDPs) are a mathematical formalization commonly applied to the reinforcement learning problem. Its usage is based on its applicability to sequential decision making, in spaces where actions influence not only the present but also future states, a common theme amongst reinforcement learning problems as previously discussed. Figure 6 describes the interaction between the agent and environment when the model is framed by such an MDP. The agent is the learner or decision-maker, and the environment is what it interacts with. These two entities continuously interact with each other, with the agent selecting an action and the environment responding to the selection via the output of the updated state and feedback on the action selection via the reward. Formally, at each time step, the agent receives a representation of the environment's internal state, based on which it then chooses an action. Upon executing this action, the environment responds by an internal change in state, observable by the agent as and a reward. This repeated interaction results in a sequence of state actions and rewards which define the MDP.[2]

The notion of a finite MDP is such that the sets of states, actions, and rewards all have a finite number of elements and so the associated random variables have well-defined probability distributions, which, in a true

sense, depend only on the previous state-action pair. Mathematically, the environment dynamics (ρ) could be given as:

$$\rho\left(s^1, r|s, a\right) = Pr\{S_t = s^1, R_t = r|S_{t-1} = s, A_{t-1} = a\}\forall s^1, s\in\delta, \quad r\in R, a\in A(s) \tag{1}$$

where the probability distribution is well-defined solely on the current and previous time steps only, satisfying the Markov property. It means that the state itself should carry with it all information of the past that may be necessary for positively influencing the future of the agent's interactions with the environment. In general, the full environment dynamics are not available and have to be learned. As Equation (1) is a probability distribution, it follows that:

$$\sum_{s\in\delta}\sum_{a\in A(s)} \rho\left(s^1, r|s, a\right) = 1\forall s^1, s\in\delta, r\in R, a\in A(s) \tag{2}$$

Returning to the concept of a value function, a metric that defines how good it is to be in a specific state, it is important to also define the notion of how good means. It is given by the return, a discounted sum of all rewards that are achieved from the current time step onwards, as illustrated in Equation (3), where $0 \le \gamma \le 1$ is known as a discount factor, and a tunable hyperparameter. The purpose of the discount factor is not only to avoid infinite cumulative rewards in nonepisodic, or never-ending, environments but also introduces the useful notion of immediate rewards being more desirable than rewards in the distant future.

$$G_t \doteq \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \tag{3}$$

For an agent to achieve the maximum return possible, it is the algorithm's job to incrementally modify the agent's policy, or its action selection process, such that its value function under the policy is maximized. Formally, if an agent follows a policy π, this is equivalent to saying it would act $A_t = a$ in state $S_t = s$ at the time t with probability π(a| s). In all reinforcement problems, there exists at least one optimal policy, π^* such that the corresponding optimal value function V^*performs better than all other nonoptimal value functions in all states.[2] By the above definition, the value function of a statesswhich follows policy$\pi\pi$is given by Equation (4), where the subscript π refers to the agent selecting actions based on policy π.

$$V_\pi(s) = E_\pi[G_t|S_t = s] = E_\pi\left[\sum_{k=0}^{\infty}\gamma^k R_{t+k+1}|S_t = s\right]\forall s\in\delta \tag{4}$$

A similar function termed the action-value function could also be defined as an expectation of the return from starting in the state s, selecting an action a, and subsequently the following policy π thereon. It could be given by:

$$\begin{aligned} Q_\pi(s,a) &= E_\pi[G_t|S_t = s, A_t = a] \\ &= E_\pi\left[\sum_{k=0}^{\infty}\gamma^k R_{t+k+1}|S_t = s, A_t = a\right] \end{aligned} \tag{5}$$

The optimal value function could then be defined as:

$$V^*(s) = maxV_\pi(s)\forall s\in\delta \tag{6}$$

Similarly, for the optimal action-value function:

$$Q^*(s,a) = max\, Q_\pi(s,a)\forall s \in \delta \& a \in A(s) \tag{7}$$

A reinforcement learning algorithm should be able to efficiently learn the optimal policy through the value function, action-value function, or even a parameterized representation of the policy directly. However, such an optimal policy could rarely be found due to the extreme computational cost required for problems of interest. For instance, tabular methods work by maintaining a table of values over the states, or state-action pairs, and this is simply intractable for larger problems due to the curse of dimensionality.[29] A solution to this is to use a function to approximate these values, where the function consists of less parameter that needs to be learned than the original problem itself. The first method to successfully address this problem is known as deep Q-network (DQN). The instability in the learning process is due to a variety of issues present in the problem set-up. The first of these is due to correlations in the sequence of observations, as each time step is fed into the network in order of observation, thereby making the data-dependent on previous time steps. This is a problem as most reinforcement learning methods, including Q-learning, assume the data to be*iid* iid(independent and identically distributed). Inexperience replay, the agent stores experiences $\varepsilon_t = (S_t, a_t, r_t, S_{t+1})$ at each time step in an overall replay memory D. In the learning stage of the algorithm, updates to the Q-function are applied to mini-batches of experiences pooled at random from this memory, removing the correlations in the observations. The second problem refers to the fact that the distribution of data in

reinforcement learning changes during the agent's learning process and its constantly updating policy. It is problematic as algorithms including Q-learning assume a fixed, nonstationary distribution. The experience replay method also helps tackle this problem by smoothing the distribution of data and preventing erratic changes on each policy update.

Algorithm The full DQN algorithm

Initialize replay memory D to capacity N.
Initialize action-value function Q with random weights
for episode =1: M do
  Receive S_1 from the environment
  for t=1:T do
    Select random action a_t with probability $\in$
    otherwise select $a_t = max\, Q^*(s_t, a; \theta)$
    $a_t = \max_a Q^*(s_t, a; \theta)$
    Execute action a_t and observe $\gamma_t r_t$ and S_{t+1}
    Store transition (S_t, a_t, r_t, S_{t+1}) in D
    Set $S_t = S_{t+1}$

$$y_j = \begin{cases} r_j & \text{for terminal } s_{j+1} \\ r_j + \gamma \max_{a_{j+1}} Q(s_{j+1}, a_{j+1}; \theta) & \text{or nonterminal } s_{j+1} \end{cases}$$

    Sample random mini batch of transitions
    (s_j, a_j, r_j, s_{j+1}) from $\mathcal{D}$
    Set: (S_j, a_j, r_j, S_{j+1})
    Perform gradient descent on $(y_j - Q(s_j, a_j; \theta))^2$

An overview of policy gradient methods will be presented, with derivations leading into the development of *PPO*, or *Proximal Policy Optimization*, the current defacto algorithm used in reinforcement learning problems. These methods parameterize the policy as $\pi(a|\, s, \boldsymbol{\theta}) = \Pr\{A_t = a|\, S_t = s, \boldsymbol{\theta}_t = \boldsymbol{\theta}\}$, where $\boldsymbol{\theta} \in \mathbb{R}^d$with d referring to the number of degrees of freedom in the parameterization of the policy. The goal of these methods is then to learn the values of $\boldsymbol{\theta}$ to shape the policy such that performance is maximized (ie, finding the optimal policy). This is done via a measure of performance $J(\boldsymbol{\theta})$ and updating the parameters via gradient ascent as in Equation (10), where $\nabla_{\boldsymbol{\theta}}\hat{J}(\boldsymbol{\theta})$ refers to a stochastic estimate with an expectation that approximates $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$.

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha \nabla_{\boldsymbol{\theta}} \hat{J}(\boldsymbol{\theta}) \tag{8}$$

Another advantage of using policy gradient methods is that the action selection changes smoothly with updates to the parameters, whereas in DQN with an $\in$-greedy policy, the action probabilities can suffer from erratic changes even after a small update to the action-value function, if such an update results in a different action having a maximal value. This property of policy gradient methods results in stronger convergence guarantees than available in DQN.[2]

In the episodic case, the performance measure is usually defined as the value functionfrom the initial starting state; $J(\boldsymbol{\theta}) \doteq v_{\pi_\theta}(s_0)$, where $v_{\pi_\theta}(s_0)$ is the true value function for the policy $\pi\boldsymbol{\theta}$. As $v_{\pi_\theta}(s_0)$ is generally not known, it is not possible to find its gradient. The policy gradient theorem handles this by rearranging it into a function whose gradient can be found. The result is shown below:

$$\nabla J(\boldsymbol{\theta}) \propto \sum_s \mu(s) \sum_a \nabla \pi(a|s) q_\pi(s,a) \tag{9}$$

Thereby providing a computable term for the gradient of the performance metric, $J(\boldsymbol{\theta})$, which does not explicitly include the state distribution. The proportionality sign in Equation (11) is sufficient as the constant of proportionality will be absorbed by the α term, a tunable hyper parameter in the update. This can then be re-written as:

$$\begin{aligned} \nabla J(\boldsymbol{\theta}) \quad &\propto \sum_s \mu(s) \sum_a \nabla \pi(a|s) q_\pi(s,a) \\ &= \mathbb{E}_\pi \left[\sum_a q_\pi(S_t, a) \nabla \pi(a|S_t, \boldsymbol{\theta}) \right] \end{aligned} \tag{10}$$

By noticing that the summation is over the states and weighted by how often those states are visited under the policy π (this is what $\mu(s)$ represents). Therefore, if the policy π is followed, the expectation above holds. This could then be used directly in the stochastic gradient ascent update, as shown in Equation (11), where $\hat{q}$ refers to a learned approximation to $q_p i$ with $\boldsymbol{\omega}$ as the parameter vector.

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha \sum_a \hat{q}(S_t, a, \mathbf{w}) \nabla \pi(a|S_t, \boldsymbol{\theta}) \tag{11}$$

However, this update involves all the actions even though only a single action was taken, again limiting the domain to a discrete action space. Instead of directly using this, the *REINFORCE* algorithm modifies the update rule to consider only A_t, the action taken at time step t. This is done via the *gradient-trick* method, as used in the policy gradient theorem derivation, giving the result below.

$$\nabla J(\boldsymbol{\theta}) = \mathbb{E}_\pi [G_t \nabla \ln \pi(A_t|S_t, \boldsymbol{\theta})] \tag{12}$$

This motivates the standard REINFORCE update rule, given by:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha G_t \nabla \ln \pi(A_t|S_t,\boldsymbol{\theta}) \tag{13}$$

which now only depends on the actual action taken, A_t This vector in the update rule points in the direction in the parameter space which will increase the probability of selecting the action A_t again, scaled by G_t, thereby forcing the policy to favor this action more if it yields a high return. A major change that can be made to this algorithm is the introduction of a baseline, $b(s)$, which does not affect the gradient as long as there is no dependence on the action.[2] A common baseline used here is $V(S_t)$. This would give the update:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha(G_t - V(S_t)) \nabla \ln \pi(A_t|S_t,\boldsymbol{\theta}) \tag{14}$$

Noting that $G_t = Q(S_t, A_t)$, the parenthesized term is equivalent to what is knownas the advantage function $\hat{A}_t$, giving:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha \hat{A}_t \nabla \ln \pi(A_t|S_t,\boldsymbol{\theta}) \tag{15}$$

This is the result for the update in the REINFORCE algorithm. However, although REINFORCE provides many benefits over the previously discussed DQN algorithm, the updates take place at the end of the episode, making it a Monte-Carlo method and therefore suffers from high variance and slow learning.[2] Defining the objective function for REINFORCE as $J^{REINFORCE}(\boldsymbol{\theta})$ that is

$$J^{REINFORCE}(\boldsymbol{\theta}) = \mathbb{E}_\pi \left[\hat{A}_t \ln \pi(A_t|S_t,\boldsymbol{\theta})\right] \tag{16}$$

Since the discovery of this algorithm, many other performance measurements havebeen proposed, including that given by Equation (17).[30]

$$J^{TRPO}(\boldsymbol{\theta}) = \mathbb{E}_\pi \left[\hat{A}_t \frac{\pi_{\boldsymbol{\theta}}(a_t|s_t)}{\pi_{\boldsymbol{\theta}_{\text{old}}}(a_t|s_t)}\right] = \mathbb{E}_\pi \left[\hat{A}_t r_t(\boldsymbol{\theta})\right] \tag{17}$$

where $\pi\boldsymbol{\theta}$ refers to the updated policy and $\pi_{\boldsymbol{\theta}_{\text{old}}}$ the policy before the update. The term *TRPO* is used for this objective, as it is the metric used in the recently developed *Trust Region Policy Method* algorithm.[31] However, this algorithm also imposes a constraint on the problem; ensure that the KL-divergence between the new and old policies is lower than a small value δ in order to ensure that the policies do not differ by *too much*. This constraint restrains the updates and offers a theoretical guarantee of policy improvements on each update for a sufficient step size. However, this method is rarely used in practice as the computation of the KL divergence via the conjugate gradient method requires the expensive calculation of the Fisher information matrix.[31-35] To combat this, the more recent collection of algorithms, known as *PPO*, approaches the problem of constraining the updates using what is known as a *clipped surrogate objective*, acting as a first order approximation to $J^{TRP\ O}(\boldsymbol{\theta})$. The new objective is given by:

$$J^{PPO}(\boldsymbol{\theta}) = \mathbb{E}_\pi \left[min\left(\hat{A}_t r_t(\boldsymbol{\theta}), \text{clip}(r_t(\boldsymbol{\theta}), 1-\in, 1+\in)\right)\right] \tag{18}$$

where, clip $(r_t(\boldsymbol{\theta}), 1 - \in, 1 + \in)$ refers to constraining the value of $r_t(\boldsymbol{\theta})$ to the interval $[r_t(\boldsymbol{\theta})- \in, r_t(\boldsymbol{\theta})+ \in]$ as an alternative method of restricting the amount by which the policy updates at each step. The paper suggests using a value of $\in \approx 0:2$. PPO is much easier to implement as it no longer requires calculation of the KL divergence, and hence neither the Fisher information matrix. It also offers another improvement upon REINFORCE; it allows multiple updates to be made per epoch, which is not possible with REINFORCE as the large updates would cause it to be unstable.[32] This tackles the original problem of REINFORCE being too slow to learn, and many empirical results have shown PPO to learn faster than simple methods such as DQN based on this and the fact that its updates can be parallelized over multiple CPUs. PPO is the current de-facto standard policy-gradient algorithm and therefore will also be used to tackle the peak-shifting task, in addition to DQN.

4 | RELATION TO THE PEAK SHIFTING DOMAIN

In the peak shifting task, there is again much flexibility in how these terms could be defined, where the chosen definitions would greatly affect how the model learns and also what the final policy would describe. It should be noted that the choice of definitions given below is not unique, and many other choices could have been made. The environment would refer to a system encompassing the battery itself, and its interaction with the outside world. That is, the weather, the electricity grid, the grid's system operator, the households, battery, solar panels, etc., are all encompassed by the notion of the environment. The agent would then refer to a non-physical CPU or the brain of the battery, which makes decisions on how much energy to buy at any single time from the grid. Following this, the action space would be a single continuous value at ≥0 and t is describing how much energy would be bought from the grid at time step t. At all points in this discussion, any reference to energy would assume units of kWh.

Upon choosing this action, the environment responds by internally calculating how much energy is required from the battery to provide the necessary energy for consumption, and, therefore, automatically performs the necessary charging and discharging of the battery without the agent needing to know how this computation works. Another obvious choice of action space here would have been to allow the agent to choose how much energy to charge or discharge from the battery. If this action space were to be used, the internal computation by the environment would instead need to calculate how much additional energy is required to provide enough energy for the consumption, and this would then be purchased from the grid. Therefore, both methods achieve the same result but via the agent to help direct its future decisions. Firstly, the agent needs to know how much energy is currently stored in the battery to make a well-informed decision on how much more energy may be required, and so the current battery charge should be included in the state definition. In addition, it is important to know how much energy is being generated and consumed by the household at the next time step. Although this may seem like cheating, that the agent knows explicitly how much energy the household would require, it should be noted that, in production, these values could be estimated through simple forecasting methods. As a single timestep is one minute, this is equivalent to using all past observational data to predict the values just one minute ahead. Based on the disaggregated data provided by the NILM technology, this forecasting should be both simple and of very high accuracy. As such, the problem of forecasting this one minute ahead data is ignored and the learning of how to optimally buy energy to approach the peak demand problem with reinforcement learning is given full attention. However, the agent still needs to learn how to effectively buy energy taking into account the seasonal nature of the data. Even though the agent has full knowledge of the next time step, it should be able to leverage information pertaining to the weather, which, if it could learn to predict, could help in its forecasting of energy generation due to photovoltaic solar cells. For this reason, all-weather data (sunlight, rainfall, etc.) would also be included in the state. Finally, the current price and demand (available energy from the grid) would also be added to the state, as this would allow the agent to include these data in its decision-making process. Finally, the total consumption of energy is then also included in the state representation. Note that this single value was chosen over the disaggregated data, but either choice would be suitable. In the effort to keep the method as general as possible, the simulator, which would be discussed in detail later, would allow for user-defined state-space representations, such as removing some of these features such as weather data or even adding in more features that could be useful for learning. The reward function needs to be defined. At this point, it is important to reiterate exactly what the aim of the task is; achieving peak shifting while reducing energy consumption and lowering utility bills, with peak shifting being of primary importance. The process of defining a suitable reward signal for this task is known as reward signal shaping. It should be noted again that there is no specific correct reward signal, and that the choice is up to the designer. For this task, individual rewards related to the various optimization tasks are first defined. In the following, the term penalty is used for negative rewards and is solely a term of convenience. In maximizing the reward, the agent is thereby minimizing these penalties. Before looking into the overall goals of peak shifting and cost and energy reduction, it is first important to look at the constraints of the system. In a standard, no reinforcement learning approach to optimization, mathematically, it might be able to formulate an objective function to maximize subject to constraints. In this problem, these constraints would be along the lines of purchase enough consumption energy, do not buy more energy than available (grid demand), and do not store more energy in the battery than permitted by its maximum capacity. In the reinforcement learning setting, these constraints cannot be enforced explicitly, but could instead shape the reward signal in such a way that the agent would be guided towards a policy that satisfies each of these. It should be noted that it is not guaranteed such a policy exists, but even if one does not, given a suitably defined reward signal, the optimal policy would be such that the constraints are violated as infrequently as possible. Before defining the various individual reward signals, a few useful terms would first be given that are internally calculated by the environment. Refers to how much energy remains in the current time step after considering the charge of the battery, how much energy has been consumed and generated, but before purchasing energy from the grid. Hence, if this value is negative, the amount by which it is below zero refers to the minimum amount of energy that the agent needs to purchase to provide enough energy for the household to continue using its appliances, with any excess being used to charge the battery. The term would be used for the agent's choice of how much energy it would purchase at the current time step.

$$\text{current\_energy} = \text{current\_charge} + \text{generated\_energy} - \text{consumed\_energy} \quad (19)$$

$$\text{min\_required\_energy} = max(0; -\text{current\_energy}) \quad (20)$$

To ensure that constraint 1 is satisfied, a penalty termed *under_purchase* is designed and given by

Equation (21), where the minus sign signifies that this is a penalty and should be avoided by the agent. If the agent were given this reward signal alone, it should learn to maximize it, or achieve the value of 0, corresponding to never purchasing too little energy

$$r_{\text{underpurchase}} = -max(0; \text{min\_required\_energy} - \text{to\_buy}) \tag{21}$$

Constraint 2 is then tackled using the penalty given by $|abovelimit$ in Equation (22), where demand refers to the current limit of available energy by the grid. If this were the only reward signal, the agent would learn to always buy less or equal to the available energy by maximizing this term to result in 0 penalties.

$$r_{\text{abovelimit}} = -max(0; \text{to\_}buy - \text{demand}) \tag{22}$$

The penalty for constraint 3, termed *abovecapacity* is given by Equation (23), where *max_capacity* refers to the maximum capacity of the installed battery. Similarly, to before, using this reward signal by itself would guide the agent into never purchasing an amount of energy that would force the battery to store more charge than is physically possible.

$$r_{\text{abovecapacity}} = -max(0; \text{current\_energy} + to\_buy - max\_\text{capacity}) \tag{23}$$

The above penalties individually tackle the constraints on the system but do no effort in explicitly working towards the intended goal of peak shifting and cost and energy consumption reduction. In order to guide the agent into buying energy at cheap prices, a penalty termed cost has been designed and is given in Equation (24), where price refers to the current price of energy, with units/kWh. This penalty should also cover the goal of reducing energy consumption as reducing the amount of energy purchased will also lower this penalty. As such, this single penalty goes some way in addressing two of the objectives posed in the peak shifting problem.

$$r_{\text{cost}} = -\text{price} \times \text{to\_buy} \tag{24}$$

The main problem of peak-shifting is not as easy to describe in terms of penalties and rewards. The previously designed penalty goes some way in addressing this by making sure the agent does not purchase more energy than available, but it does not explicitly tell the agent to smooth its load profile, that is buy similar amounts of energy at each time step. To address this explicitly, the reward termed has been designed and is given in Equation (25). This positive reward has a maximum value of 1 when the agent chooses to not buy any energy, and this decreases linearly towards 0 as the amount of energy purchased approaches the demand limit. This does not explicitly tell the agent to favor purchasing similar amounts of energy at each time step but tries to keep the chosen amount as close to 0 at each time step. As discussed previously, there are an infinite number of rewards that could be designed to tackle the same or similar problems. The purpose of this task is not to find the best reward function to address the problem, but to verify that reinforcement learning is a suitable solution method to this task, and as such, it is not necessary to investigate defining overly intricate reward functions.

$$r_{\text{belowlimit}} = max\left(0, 1 - \frac{\text{to\_buy}}{\text{demand}}\right) \tag{25}$$

It should be noted that these rewards may not be mutually exclusive. For example, *abovelimit* and *belowlimit* both address very similar problems to do with influencing the amount of energy purchased with respect to how much is available. This observation means that there may be some interaction between the different terms upon combining the individual rewards and penalties. These five individual signals are then linearly combined, as shown in Equation (26), to provide a single overall reward signal to guide the agent's learning process. The values of the α_i coefficients will be discussed at a later stage. It will be seen later in the results and evaluation section that the appropriate selection of these values is paramount to the success of the agent learning an optimal policy and is therefore one of the main contributions presented in this work.

$$r = \alpha_1 r_{\text{underpurchase}} + \alpha_2 r_{\text{abovelimit}} + \alpha_3 r_{\text{abovecapacity}} + \alpha_4 r_{\text{cost}} + \alpha_5 r_{\text{belowlimit}} \tag{26}$$

5 | RESULTS AND DISCUSSIONS

All models have been trained using 70% of the data for household A, and the results that follow have been generated using the remaining 30% of the data for the same household, corresponding to approximately two months. This train/test split is used to ensure that the models do not overfit to the data used for training but instead generalizes to unseen data and therefore could be used in production where future data is of course not readily available. All of the graphs and results in this section have been generated automatically using the simulator, at the same time that the model has been trained.

In order to be able to verify whether or not the reinforcement learning agents have learned a useful policy, it is necessary to first construct a simple baseline with which results could be compared against. This baseline should follow as good a policy as is possible without using reinforcement learning to ensure that the comparisons are meaningful. The baseline has been chosen to follow a policy that requests exactly the amount of energy it needs at every time step and therefore does not require the use of a battery. To this end, the baseline has an instant advantage over all reinforcement learning ends in that it does not need to learn to infer how much energy is needed at any time. The calculation is done internally, taking into account energy consumption and generation, given by Equation (27), taking into account that there is no battery and hence charge is removed from this equation

$$required\_energy = consumed\_energy - generated\_energy \tag{27}$$

As the baseline simply purchases what it needs at the time, there would be times when it tries to purchase more energy than is currently available. This is the heart of the problem and is exactly why the reinforcement learning agents would need to learn how to peak-shift. Although there is no training necessary for the baseline model, in order to allow comparison with the results from the reinforcement learning agents, the following graphs and results are generated using just the 30% testing data. Figure 7 gives the requested energy profile for the baseline agent. This, along with all other graphs that follow, presents the averaged data over a 24-hour period. The shadows behind each curve show a single SD either side of the mean in order to visualize the variance in the data. As the actions that the agent takes are the decision of how much energy to buy, this graph is a direct visualization of the agent's policy. As seen from Figure 8, the peak is not as apparent as that seen in the original household data, which included all the training data. Hence, this baseline agent does not attempt to purchase more energy than available too frequently, but the purpose of the task is also to achieve peak shifting in general and to flatten the requested energy profile as much as possible while taking into account the other objectives of lowering utility bills and reducing overall energy consumption.

It should be noted again that the baseline does not use any control; it purchases exactly what the amount of energy it needs at every time step, and hence does not take into account the price or demand (limit) of energy at all. Figure 11 shows the same requested energy profile

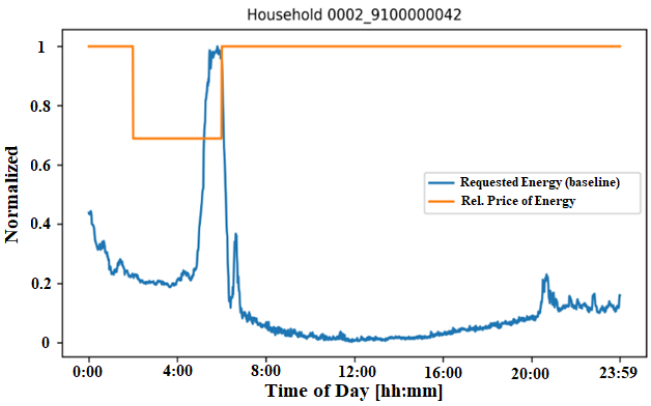

FIGURE 8 Daily-averaged normalized pricing data for the baseline model on household A [Colour figure can be viewed at wileyonlinelibrary.com]

TABLE 3 Numerical results for the baseline agent

Parameter	Results
Error *abovecapacity* abovecapacity	0 (0.00%)
Error *underpurchase* underpurchase	0 (0.00%)
Error *abovelimit* abovelimit	122 (0.15%)
Monthly bill	3517.79 Unit price
Total energy purchased	292.51 kWh

but now graphed against the price of energy and normalized. This allows the designer to verify if the agent has attempted to buy more energy when the prices are lower. Of course, in the baseline model, there is no relationship between the two curves as the agent does not take the pricing data into account. However, by coincidence, the cheaper band of energy lines up with the peak in the requested energy profile and for this reason, the baseline agent luckily has purchased a lot of its energy at the cheapest possible price.

It should be noted that although the baseline may seem simple at first, the fact that it does not need to learn how much energy is required at any time, and also that it purchases a lot of its energy at the cheapest price possible makes it a very strong baseline to outperform. Table 3 gives the results for the baseline agent on the test data. The *abovecapacity* error refers to the number of times the agent attempted to charge the battery above its maximum capacity. As the baseline agent does not use a battery, this error. *underpurchase* refers to the agent buying less energy than is required at that time step. As the definition of the baseline policy is such that it would always buy exactly as much as it needs, again this is never encountered. Finally, the error refers to the number of times that the agent attempted to purchase more energy

than was available by the grid. As a single time step corresponds to one minute, the baseline agent would have left this household without the required electrical energy for two hours, corresponding to 0.15% of the test data period. For other households, this number is much higher, and so the reinforcement learning agent needs to outperform an agent which itself is already quite strong in the case of household A.

The first reinforcement learning algorithm that was used was DQN based on its simplicity and ease of training. This allowed for many models to be trained and is how the initial choice of individual reward signals was chosen. This was based on first trying one reward signal, and then subsequently adding the rest whilst observing the resultant model's policy. At this stage, it became very apparent that due to the large difference in magnitudes of the individual reward signals, without the automatic penalty shaping method none of the models were able to learn a useful policy. To this end, as discussed in the Simulator section, once a model has trained the coefficients in the linear combination of the individual signals were automatically scaled in order to improve the next model's policy. The discretization of the action-space was varied in the initial models, and [0; 0:005; 0:01; 0:015; 0:02; 0:03; 0:04] was chosen to be used in all future models. This choice was based on a number of factors as well as from the initial results that proved this to be a promising set of actions. Firstly, keeping the action space constant throughout the models allows comparisons to be made much more easily. Secondly, on observation of the requested energy profile for the baseline agent in Figure 10, it was seen that the maximum required energy at a single time step hovers around the 0:03-0:04 range, and so this was chosen as the highest values in the discretization. Also, as most time steps require much less energy, it was chosen to use more actions between 0:00 and 0:02. All of these values have units of kWh. Increasing the number of actions made the training much slower and so this upper limit of 7 actions was kept final. In addition to the varying reward signal coefficient values, the episode length and neural network architecture were also varied on each model. For each model, an episode length was chosen at random from 1440 and 10 080, corresponding to 1 day and 1 week, respectively. The model architecture was also chosen at random between [64], [64; 64], and [64; 128; 64], where each value represents the number of nodes in each layer. It should be noted here that the variation in these two parameters would most likely also affect the optimal values for the reward signal coefficients, and this is one downside to the simple scaling method used. In future work, a more intricate penalty shaping method could be used which takes into account all past results and the model architectures. The best model was selected after training approximately 300 models using the created simulator. As previously discussed, there is no explicit definition of what best here means, and so the author's judgment has been used but ensuring that peak shifting maintains the number one priority with a reduction in energy consumption and lowering of utility bills also being important. Table 4 gives the results for this model on the test data and is compared side-by-side with the baseline results from the previous section. Now that there is a battery, it is possible to encounter the error. The DQN model attempts to overcharge the battery approximately 13% of the time. This is a significantly high amount, but it should also be noted that most models encountered this error at least 25% of the time. Although high, in production it is possible to explicitly constrain the amount of energy bought so as not to overcharge the battery, and so the error is not of too much concern. However, the fact that the model has relied on overcharging the battery at some time steps implies that the policy it has learned is not optimal. One way of tackling this in the future would be to explicitly give the battery's maximum capacity as an additional feature in the environment's state representation. The DQN agent has also encountered the error at 512-time steps, corresponding to not buying enough energy for the required consumption. However, as this is only 0:65% of the time, this implies the agent has almost perfectly learned how to infer the required energy, making a mistake very infrequently. The times at which it could be attributed to the coarse discretization of the action space. For example, consider the case where the policy in a specific state which requires 0:025 kWh of energy assigns 49% chance of purchasing 0:03 kWh and a 51% chance of purchasing 0:02

TABLE 4 Results for the best DQN model compared to the baseline

Parameter	Baseline Agent	DQN Agent	% Difference
Error *abovecapacity*	-	10 374 (13.10%)	-
Error underpurchase *underpurchase*	0 (0.00%)	512 (0.65%)	$+\infty$ %
Error abovelimit *abovelimit*	122 (0.15%)	29 (0.04%)	−76%
Monthly bill	3517.79 UC	3296.43 UC	−6%
Total energy purchased	292.51 kWh	257.94 kWh	−12%

kWh of energy. In this case, the agent would, but if there were a less coarse discretization, with some actions in between 0:02 and 0:03, then the agent would be likely to select a more appropriate action. This alone is a motivation for using a continuous-action method such as PPO. However, the error has been reduced by 76% compared to the baseline, which is evidence that the agent has learned to peak shifts considerably. It is still purchasing more than the limit at some time steps, but there has been considerable improvement in this domain. Finally, the monthly electricity bill has had a small reduction by 6% and energy consumption has also reduced by 12%. These results prove that the DQN agent has outperformed the baseline and that it has successfully achieved all three of the goals in the multi-objective optimization problem.

Figure 9 visualizes the policy with control (DQN agent) and compares it to that of the baseline, providing verification that peak-shifting has successfully occurred. There is quite a bit of fluctuation in the DQN model's requested energy profile, but this is due to the discrete nature of the action space. As the PPO agent is able to work in continuous action space, its resultant profile should appear to be a lot smoother. There is a slight peak in the purchase of energy between 2 AM and 6 AM

Observing Figure 10, which shows the daily-averaged normalized values of the requested energy profile against the price of energy, it could be seen that this peak corresponds to a time period where energy is cheaper. Therefore, evidence that the agent has learned to buy energy when it is cheaper in an effort to reduce the electricity bill. Figure 11 shows the daily-averaged charging profile of the battery over the test data when using this model, an indirect method of visualizing the policy as the battery charge is related to the amount of energy that is purchased. As the battery that has been tested has a maximum capacity of 14.4 kWh, it could be seen that the relatively high charge state of the battery between noon and 8 PM most likely corresponds to the numerous errors encountered.

The peak between 2 AM and 6 AM corresponds to the agent purchasing more energy at this time due to the lower price, with the excess being stored in the battery. The sharp loss of charge just after this time then corresponds to the necessary use of these stores to tackle the large consumption period as given by the initial peak before implementing the battery. The relatively high amount of energy being stored thereafter corresponds to the agent purchasing energy even though consumption is relatively low, and so the majority of this purchased energy is being stored in the battery. It could also be due to it requiring more energy the next day quite early on and so the agent has realized this and is purchasing the energy well in advance. This is corroborated by the fact that it then begins to use this energy continuously from about 4 PM until 2 AM the next day when it again decides

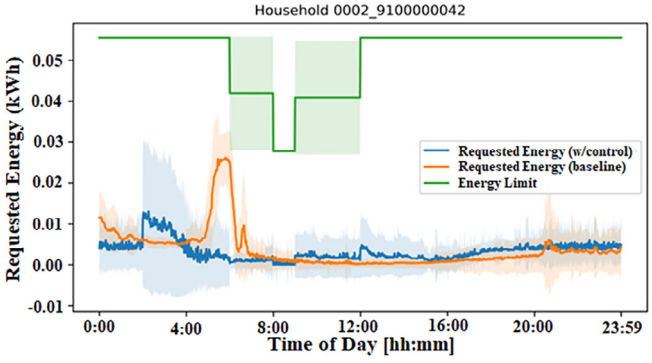

FIGURE 9 Daily-averaged requested an energy profile for the best DQN model on household A [Colour figure can be viewed at wileyonlinelibrary.com]

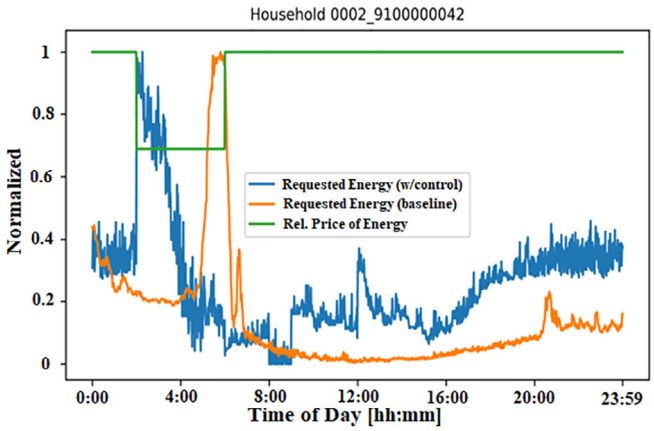

FIGURE 10 Daily-averaged normalized pricing data for the DQN model on household A [Colour figure can be viewed at wileyonlinelibrary.com]

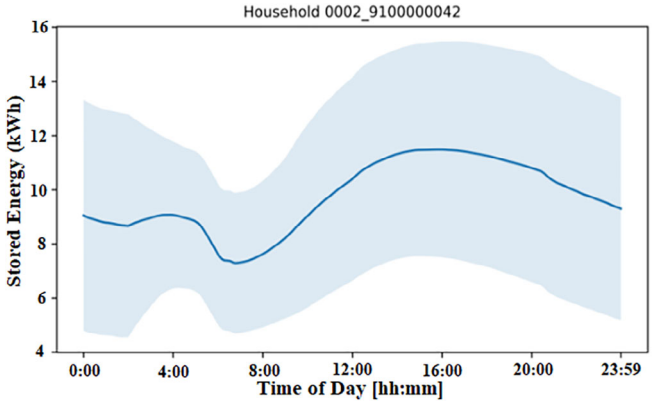

FIGURE 11 Daily-averaged normalized charging profile for the DQN model on household A [Colour figure can be viewed at wileyonlinelibrary.com]

to top up the energy in advance of the high consumption period.

Figure 12 shows the total reward achieved per episode. The fluctuation could be attributed to the fact that each episode corresponds to a single day of data, and the day-to-day data varies significantly as seen by the large error shadow in the household data as previously seen in Figure 4. However, the model could be seen to learn very quickly by the initial upwards trend, and learning slows by around episode 75. All 1-day episode models were trained for 500 episodes to allow for sufficient exploration of the domain before stopping. At the end of the training process, the model from around episode 75 would have been used to avoid any overfitting that may have occurred between that and the end episode.

It should be noted that the ability of the agent to successfully achieve the goals of the task was highly reliant upon it learning from an optimized reward signal, which was shaped using the automatic penalty shaping feature discussed previously. Figure 13a shows the individual reward signals per episode during training for the first DQN model that was trained. This model does not use the automatic penalty shaping feature as there is no previous model from which to learn. It could be seen that the reward, the reward is given for being below the available energy limit, is much higher than the rest. Such a reward signal would bias the agent into placing more emphasis on learning to maximize this reward whilst ignoring the rest. Figure 13b then shows the same graph, but for the best DQN model that has been evaluated in this section. It could be seen that, through the automatic penalty shaping feature, the magnitudes of each individual signal are much closer to each other. The only anomaly now is the erratic fluctuation in the penalty. This fluctuation corresponds to the large error achieved by this agent. This implies that if there were a way to smoothen this specific reward signal, then the model agent would be more likely to learn a policy that avoids such a large number of errors related to the battery capacity. In summary, the DQN results are very promising, having achieved all three intended results of peak-shifting, lower energy bills, and reduced energy consumption. There is, however, still room for improvement in regard to the aforementioned errors. To this end, it is worth mentioning again that although it has achieved considerable peak shifting, it is not perfect and still purchases more energy than is available sometimes, albeit very infrequently. The main functional difference between DQN and PPO is that PPO could handle a continuous action space and so does not suffer the same problems as DQN does by discretization. PPO learns to modify the mean and variance of a Gaussian which is then used in the action selection.

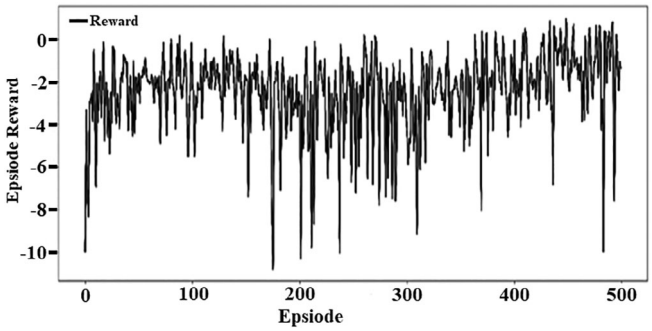

FIGURE 12 Total rewards per episode during the training period for the best DQN model

As Gaussians have an infinite domain, this is then clipped to [−1; 1] and then scaled appropriately to [0; 0:03], as suggested in the literature. This modified range is chosen for the same reasons as with the DQN considerations based on the observed required energy from Figure 7. Similarly, when training the DQN models, the parameters of the model were adjusted over the first few initial models until a parameter set that performed well was found.

All models were trained using episodes of 1 day (1440-time steps). The agents did not seem to learn efficiently when using an episode length of one week with the chosen parameters, and it was decided that finding parameters for a one-week episode model would be unnecessary additional extra work given the 1-day episode models were training easily and more quickly. More than 75 models

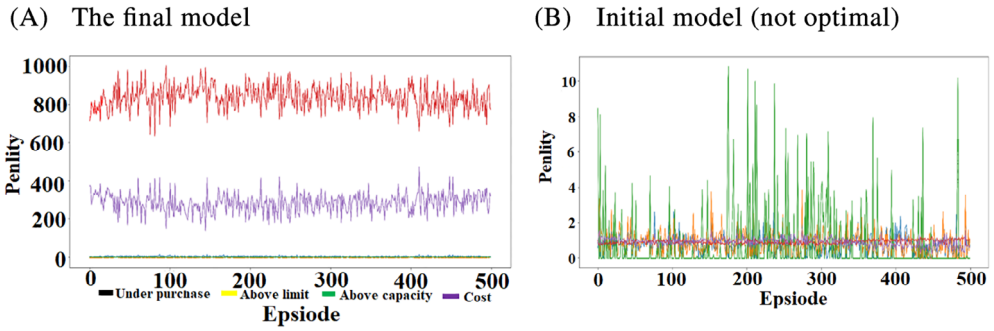

FIGURE 13 Disaggregated reward signals for two different DQN models [Colour figure can be viewed at wileyonlinelibrary.com]

Parameter	Baseline Agent	DQN Agent	% Difference
Error *abovecapacity*abovecapacity	-	5928 (7.48%)	-
Error underpurchase *underpurchase*	0 (0.00%)	0 (0.00%)	+0%
Error abovelimit *abovelimit*	122 (0.15%)	0 (0.00%)	−100%
Monthly bill	3517.79 UC	2777.75 UC	−21%
Total energy purchased	292.51 kWh	226.49 kWh	−23%

TABLE 5 Results for the best model compared to the baseline

using a PPO agent were trained, with each successive model learning from the last via the automatic penalty shaping feature. The results for this model have been compared with the baseline in Table 5. As before, the model tends to try and charge the battery above its maximum capacity from time-to-time, but much less so than the DQN model did, reducing this to 7.48% of the time compared to 13.1% as a major improvement. In addition to this, the model never or goes above the limit, that is, it always has at least enough energy than is required for the consumption and never goes above the amount of energy available by the grid. Hence, it has achieved perfect peak-shifting, the main requirement for this task. Finally, it has also managed to reduce the total energy bill and energy consumption by over 20% in both cases, thereby making this model production-ready and a perfect validation that reinforcement learning is an adequate solution to this task.

Figure 14 shows the requested energy profile of this model against the baseline. It follows a similar trend to that of DQN, but with a much smoother profile due to the ability to choose actions from a continuous space. Again, it is immediately obvious from this graph that peak-shifting has been achieved; the requested energy profile for the PPO model maintains a relatively flat profile without any significant peaks. It has also learned to react significantly to changes in the demand (available energy) profile as could be seen just after 8 AM At this time, the energy limit drops very slightly, and the model reacts to this by purchasing no energy at all, due to its prediction that the limit would rise again shortly.

Figure 15 shows this data against the price of energy after being normalized. Similarly, to the DQN model, the PPO agent decides to purchase more energy between 2 AM and 6 AM when the price of energy is at its lowest. It also seems to predict the end of this low pricing period and decides to purchase a large amount of energy again just before the price goes up, in order to minimize the overall energy bill.

Figure 16 shows the charge profile for this model, again showing a similar trend to that of the DQN model and reacting to the households' energy consumption habits almost identically. Figure 17 shows the total

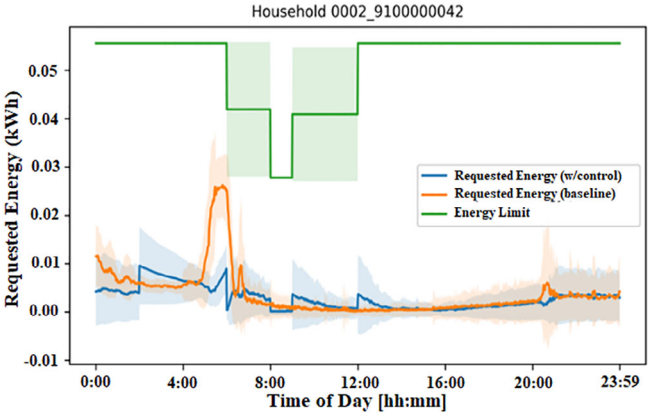

FIGURE 14 Daily-averaged requested an energy profile for the best PPO model on household A [Colour figure can be viewed at wileyonlinelibrary.com]

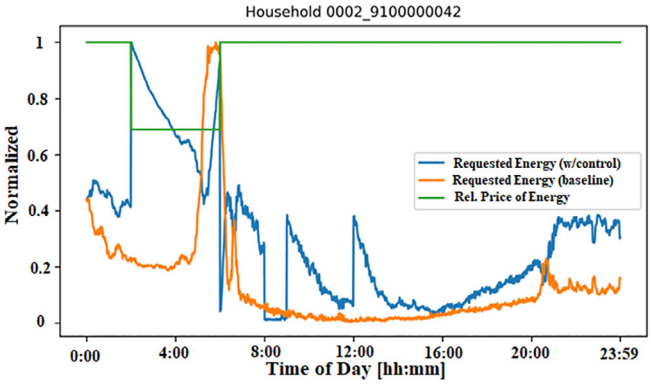

FIGURE 15 Daily-averaged normalized pricing data for the PPO model on household A [Colour figure can be viewed at wileyonlinelibrary.com]

reward achieved per episode. There is still fluctuation in this learning process due to the variation in consumption data from day-to-day. The model was only trained for 75 episodes based on the fact that it was trained over 8 different CPUs and so this equates to much more clock-training time than in the DQN model, although due to the different learning processes these cannot be directly compared anyway. The model appears to continue learning until the end of the 75 episodes, but due to memory

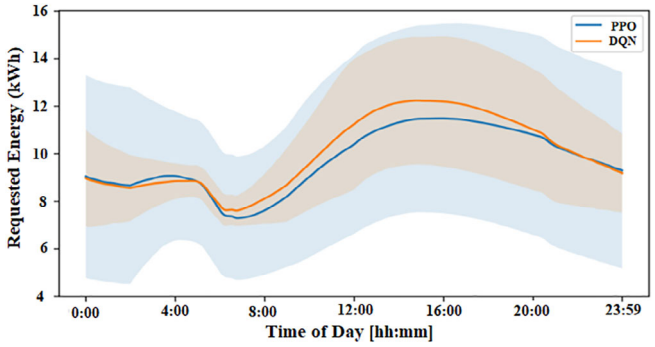

FIGURE 16 Daily-averaged normalized charging profile for the PPO model on household A [Colour figure can be viewed at wileyonlinelibrary.com]

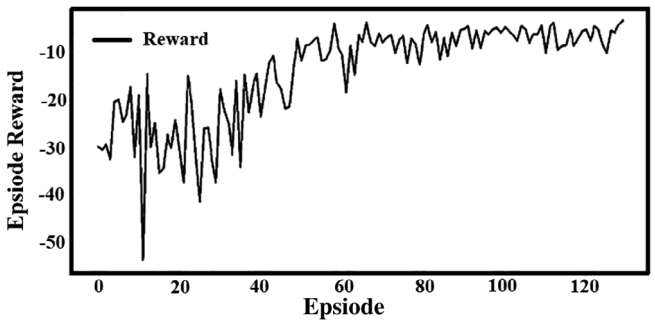

FIGURE 17 Total reward per episode during the training period for the best PPO model

reasons, it became a lot more difficult to train any further than this, although it may have proven beneficial if possible.

Figures 18a, b, similarly to the DQN case, shows how the individual reward signals have been shaped from the initial model to the final, best model chosen. Again, it is very clear that in the first model before any smart shaping has taken place, the reward overpowers the rest and so the model ignores the remaining reward signals. In the final, improved model, the reward signals are much closer to each other, although there is still some fluctuation in the above capacity reward, albeit much less than in the DQN case. However, this fluctuation begins to die off near to the end of the training period, resulting in a much lower above capacity error of approximately 7% compared to 13% with DQN. In summary, PPO gives perfect results minus its reliance on going above the battery's capacity approximately 7% of the time. It has achieved perfect peak shifting, never purchases less energy than is required, and has reduced energy consumption and lowered the utility bill by a significant amount. Although it is obvious that the PPO agent has managed to improve upon the DQN agent on all fronts, it serves useful to

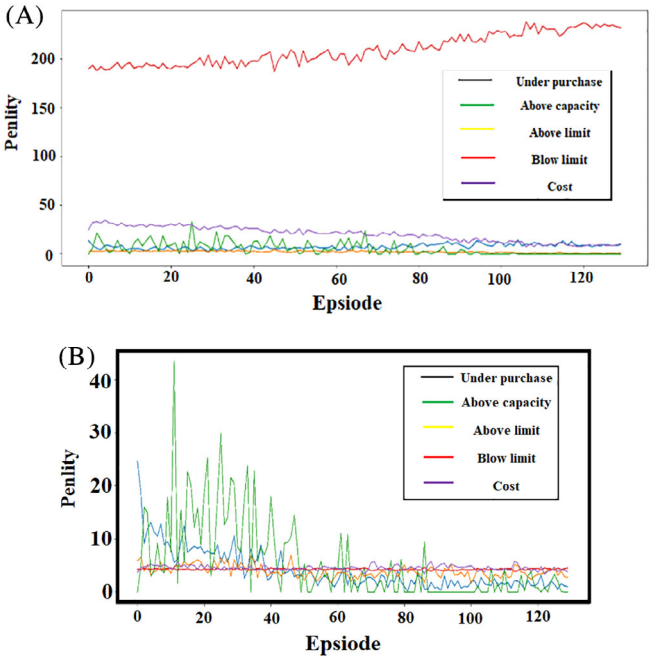

FIGURE 18 Disaggregated reward signals for two different PPO models [Colour figure can be viewed at wileyonlinelibrary.com]

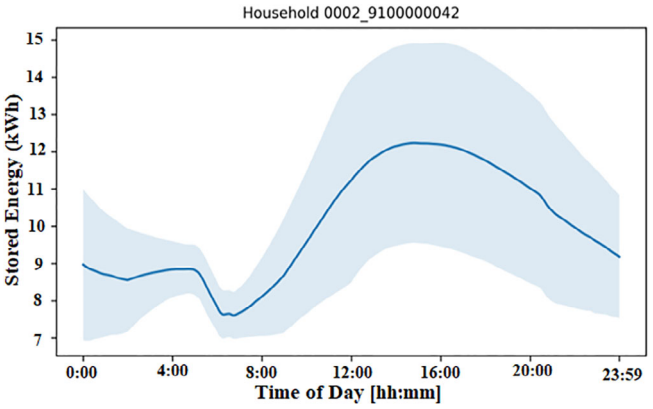

FIGURE 19 Daily-averaged requested an energy profile for the best PPO and DQN model on household A [Colour figure can be viewed at wileyonlinelibrary.com]

compare the policies on both fronts to see exactly what was learned. Figure 19 shows the requested energy profile, the direct policy, of both the best PPO and DQN agents against the baseline.

Figure 20 does the same but for the charge profile, which is an indirect visualization of the policy. From both of these graphs, it is apparent that the learned policies for both agents are incredibly similar. Also, it could be seen that the variance in action selection is much lower for the PPO agent, due to the continuous action

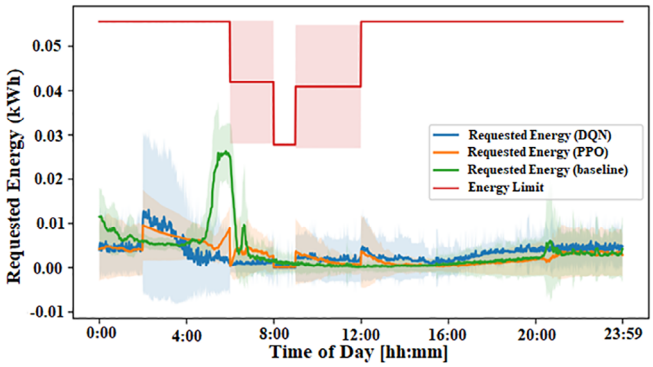

FIGURE 20 Daily-averaged requested an energy profile for the best PPO and DQN model on household A [Colour figure can be viewed at wileyonlinelibrary.com]

space, and this results in much smoother profiles and is also likely the reason why PPO never purchases more energy than is available.

The results of PPO fare better than the DQN model, and this is again undoubtedly due to its use of continuous action space and the fact that it is a more robust reinforcement learning algorithm in general, being the first choice for many problems to date. Although it is impossible to tell what the actual optimal policy in this domain is, from these results and the fact that both agents' policies are very similar, it is very likely that the policy learned by PPO is exceedingly close to the optimal policy, if not the optimal policy itself. It is hypothesized that the variation in results between DQN and PPO would disappear if the discretization of the action space in the DQN model was made infinite, approaching that of the continuous domain. Of course, this is not possible to test due to computational reasons, but the sheer similarity in the outlined policies for both agents in these graphs back up this statement.

6 | CONCLUSIONS

The aim of this paper was to verify that reinforcement learning is a suitable solution method for the peak demand problem. That is, reinforcement learning could be used in conjunction with a BSS to purchase energy at off-peak periods in order to flatten the energy requirement profile of consumers. Such an achievement would prevent the grid's system operators from needing to use peaked plants to provide additional energy during peak periods, lowering carbon emissions, and energy prices for the consumer. This peak-shifting would allow the grid's system operators to be able to more easily predict electricity demand, thereby reducing their need to generate more energy than necessary, again lowering the tariffs for energy for the consumer. Secondary aims of directly reducing the energy consumption and utility bills were also required, making this a multi-objective optimization problem. The user data, in conjunction with the created simulator which performs the full training and testing phases of the models, to find an optimal policy using DQN and PPO reinforcement learning algorithms. Finally, the proposed algorithm is able to achieve perfect peak shifting, a reduction in the monthly utility bill by 21%, and also a reduction in energy consumption by 23%, achieving all of the aims of the paper.

ORCID

Mostafa Al-Gablawy https://orcid.org/0000-0003-0231-6618

REFERENCES

1. Haider HT, See OH, Elmenreich W. A review of residential demand response of smart grid. *Renew Sustain Energy Rev*. 2016;59:166-178.
2. Yao E, Samadi P, Wong VWS, Schober R. Residential demand side management under high penetration of rooftop photovoltaic units. *IEEE Trans Smart Grid*. 2016;7(3):1597-1608.
3. Wang R, Wang P, Xiao G, Gong S. Power demand and supply management in microgrids with uncertainties of renewable energies. *Int J Electr Power Energy Syst*. 2014;63:260-269.
4. Liu Y, Yuen C, Huang S, Ul Hassan N, Wang X, Xie S. Peak-to-average ratio constrained demand-side management with consumer's preference in residential smart grid. *IEEE J Sel Top Signal Process*. 2014;8(6):1084-1097.
5. Cocco D, Serra F, Tola V. Assessment of energy and economic benefits arising from syngas storage in IGCC power plants. *Energy*. 2013;58:635-643.
6. Pascual J, Barricarte J, Sanchis P, Marroyo L. Energy management strategy for a renewable-based residential microgrid with generation and demand forecasting. *Appl Energy*. 2015;158:12-25.
7. Tian W, Heo Y, De Wilde P, et al. A review of uncertainty analysis in building energy assessment. *Renew Sustain Energy Rev*. 2018;93:285-301.
8. Shi Y, Xu B, Wang D, Zhang B. Using battery storage for peak shaving and frequency regulation: joint optimization for superlinear gains. *IEEE Trans Power Syst*. 2018;33(3):2882-2894.
9. Setlhaolo D, Xia X. Optimal scheduling of household appliances with a battery storage system and coordination. *Energ Buildings*. 2015;94:61-70.
10. Hashim H, Ho WS, Lim JS, Macchietto S. Integrated biomass and solar town: Incorporation of load shifting and energy storage. *Energy*. 2014;75:31-39.
11. Barzin R, Chen JJJ, Young BR, Farid MM. Peak load shifting with energy storage and price-based control system. *Energy*. 2015;92:505-514.
12. Uddin M, Romlie MF, Abdullah MF, Abd Halim S, Abu Bakar AH, Chia Kwang T. A review on peak load shaving strategies. *Renew Sustain Energy Rev*. 2018;82:3323-3332.
13. Mohammad Rozali NE, Ho WS, Wan Alwi SR, et al. Peak-off-peak load shifting for optimal storage sizing in hybrid power

systems using Power Pinch Analysis considering energy losses. *Energy*. 2018;156:299-310.
14. Taylor Z, Akhavan-Hejazi H, Cortez E, et al. Customer-side SCADA-assisted large battery operation optimization for distribution feeder peak load shaving. *IEEE Trans Smart Grid*. 2019;10(1):992-1004.
15. Ananda-Rao K, Ali R, Taniselass S. Battery energy storage system assessment in a designed battery controller for load leveling and peak shaving applications. *J Renew Sustain Energy*. 2017;9(4):044107.
16. Bao G, Lu C, Yuan Z, Lu Z. Battery energy storage system load shifting control based on real-time load forecast and dynamic programming. *2012 IEEE International Conference on Automation Science and Engineering (CASE)*. Seoul, South Korea: IEEE; 2012:815-820.
17. Qin J, Chow Y, Yang J, Rajagopal R. Online modified greedy algorithm for storage control under uncertainty. *IEEE Trans Power Syst*. 2016;31(3):1729-1743.
18. Kim S, Lim H. Reinforcement learning based energy management algorithm for smart energy buildings. *Energies*. 2018;11(8):2010.
19. Bui V-H, Hussain A, Kim H-M. Double deep Q -learning-based distributed operation of battery energy storage system considering uncertainties. *IEEE Trans Smart Grid*. 2020;11(1):457-469.
20. Shen S, Sadoughi M, Chen X, Hong M, Hu C. A deep learning method for online capacity estimation of lithium-ion batteries. *J Energy Storage*. 2019;25:100817.
21. Ren L, Zhao L, Hong S, Zhao S, Wang H, Zhang L. Remaining useful life prediction for lithium-ion battery: a deep learning approach. *IEEE Access*. 2018;6:50587-50598.
22. Meng J, Cai L, Stroe D-I, Ma J, Luo G, Teodorescu R. An optimized ensemble learning framework for lithium-ion battery state of health estimation in energy storage system. *Energy*. 2020;206:118140.
23. Shen S, Sadoughi M, Li M, Wang Z, Hu C. Deep convolutional neural networks with ensemble learning and transfer learning for capacity estimation of lithium-ion batteries. *Appl Energy*. 2020;260:114296.
24. Bhowmik A, Castelli IE, Garcia-Lastra JM, Jørgensen PB, Winther O, Vegge T. A perspective on inverse design of battery interphases using multi-scale modelling, experiments and generative deep learning. *Energy Storage Mater*. 2019;21:446-456.
25. Yang JJ, Yang M, Wang MX, Du PJ, Yu YX. A deep reinforcement learning method for managing wind farm uncertainties through energy storage system control and external reserve purchasing. *Int J Electr Power Energy Syst*. 2020;119:105928.
26. Chemali E, Kollmeyer PJ, Preindl M, Emadi A. State-of-charge estimation of Li-ion batteries using deep neural networks: A machine learning approach. *J Power Sources*. 2018;400:242-255.
27. Liu Y, Guo B, Zou X, Li Y, Shi S. Machine learning assisted materials design and discovery for rechargeable batteries. *Energy Storage Mater*. 2020;31:434–450.
28. Sedighizadeh M, Esmailib M, Jamshidia A. M. Ghaderia Stochastic multi-objective economic-environmental energy and reserve scheduling of microgrids considering battery energy storage system. *Electr Power Energy Syst*. 2019;106:1-16.
29. Sedighizadeh M, Shaghaghi-shahr G, Esmaili M, Aghamohammadi M. Optimal distribution feeder reconfiguration and generation scheduling for microgrid day-ahead operation in the presence of electric vehicles considering uncertainties. *J Energy Storage*. 2019;21:58-71.
30. Meng F, Zeng X-J, Zhang Y, Dent CJ, Gong D. An integrated optimization + learning approach to optimal dynamic pricing for the retailer with multi-type customers in smart grids. *Inf Sci (Ny)*. 2018;448–449:215-232.
31. Arulkumaran K, Deisenroth MP, Brundage M, Bharath AA. Deep reinforcement learning: a brief survey. *IEEE Signal Process Mag*. 2017;34(6):26-38.
32. Kakade S, Langford J. Approximately optimal approximate reinforcement learning. *Proc 19th Int Conf Mach Learn*, Vol. 2. Sydney, Australia: International Machine Learning Society; 2002:267-274.
33. Schulman J, Levine S, Moritz P, Jordan MI, Abbeel P. Trust region policy optimization. *32nd Int. Conf. Mach. Learn. ICML*, Vol. 3. France: International Conference on Machine Learning; 2015;2015:1889-1897.
34. Tang CY, Liu CH, Chen WK, You SD. Implementing action mask in proximal policy optimization (PPO) algorithm. *ICT Express*. 2020;6(3):200-203.
35. Al-Gabalawy M. Machine learning for aircraft control. *J Adv Res Dyn Control Syst*. 2019;11:3165-3191.

How to cite this article: Al-Gablawy M. Optimal peak shifting of a domestic load connected to utility grid using storage battery based on deep Q-learning network. *Int J Energy Res*. 2020;1–19. https://doi.org/10.1002/er.6023