Title

Controlling distributed energy resources via deep reinforcement learning for load flexibility and energy efficiency

Authors

Touzani, Samir
Prakash, Anand Krishnan
Wang, Zhe
et al.

Publication Date

2021-12-01

DOI

10.1016/j.apenergy.2021.117733

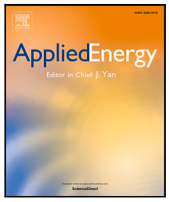
Peer reviewed

Applied Energy 304 (2021) 117733

Contents lists available at ScienceDirect

Applied Energy

journal homepage: www.elsevier.com/locate/apenergy



Controlling distributed energy resources via deep reinforcement learning for load flexibility and energy efficiency

Samir Touzani[1], Anand Krishnan Prakash[1], Zhe Wang[1], Shreya Agarwal, Marco Pritoni *, Mariam Kiran, Richard Brown, Jessica Granderson

Lawrence Berkeley National Laboratory, United States of America

ARTICLE INFO

Keywords:
Deep reinforcement learning
Deep deterministic policy gradient algorithm
Smart buildings
Control systems
Distributed energy resources
Load flexibility
Energy efficiency

ABSTRACT

Behind-the-meter distributed energy resources (DERs), including building solar photovoltaic (PV) technology and electric battery storage, are increasingly being considered as solutions to support carbon reduction goals and increase grid reliability and resiliency. However, dynamic control of these resources in concert with traditional building loads, to effect efficiency and demand flexibility, is not yet commonplace in commercial control products. Traditional rule-based control algorithms do not offer integrated closed-loop control to optimize across systems, and most often, PV and battery systems are operated for energy arbitrage and demand charge management, and not for the provision of grid services. More advanced control approaches, such as MPC control have not been widely adopted in industry because they require significant expertise to develop and deploy. Recent advances in deep reinforcement learning (DRL) offer a promising option to optimize the operation of DER systems and building loads with reduced setup effort. However, there are limited studies that evaluate the efficacy of these methods to control multiple building subsystems simultaneously. Additionally, most of the research has been conducted in simulated environments as opposed to real buildings. This paper proposes a DRL approach that uses a deep deterministic policy gradient algorithm for integrated control of HVAC and electric battery storage systems in the presence of on-site PV generation. The DRL algorithm, trained on synthetic data, was deployed in a physical test building and evaluated against a baseline that uses the current best-in-class rule-based control strategies. Performance in delivering energy efficiency, load shift, and load shed was tested using price-based signals. The results showed that the DRL-based controller can produce cost savings of up to 39.6% as compared to the baseline controller, while maintaining similar thermal comfort in the building. The project team has also integrated the simulation components developed during this work as an OpenAIGym environment and made it publicly available so that prospective DRL researchers can leverage this environment to evaluate alternate DRL algorithms.

1. Introduction

Behind-the-meter distributed energy resources (DERs) include local electricity generation (e.g., solar photovoltaic [PV]), energy storage devices (e.g., battery units), and thermostatically controlled loads (TCLs) such as heating, ventilation and air conditioning systems (HVAC) [1]. DERs are transforming the demand side of the grid from traditionally passive to active systems [2] and they are increasingly being considered as solutions to support carbon emissions and energy saving goals, and to maximize grid reliability and resiliency [3]. However, coordinating these resources with traditional building loads to improve load flexibility and grid response is not common practice in commercial control systems [4]. The challenge is how to implement a DER energy management system such that the control strategy can fulfill the grid flexibility commitment without affecting building services (e.g., occupant comfort) [3].

Traditional rule-based control algorithms do not offer integrated control to coordinate across systems and they typically operate sub-optimally [4]. On the other hand, several advanced control algorithms have been proposed in academia in the last two decades, including Model Predictive Control [5–7], Particle Swarm Optimization [8,9], Neural Networks [10,11] for similar energy management problems. Nevertheless, these algorithms have not been adopted by the controls industry at scale, yet. For instance, model-predictive control (MPC) is currently the state of the art for building controls, particularly those

* Corresponding author.
E-mail address: mpritoni@lbl.gov (M. Pritoni).
[1] These authors contributed equally to this work.

https://doi.org/10.1016/j.apenergy.2021.117733
Received 12 March 2021; Received in revised form 17 August 2021; Accepted 26 August 2021
Available online 13 September 2021

Nomenclature

$\$_{saving}(\%)$	Cost savings %
$\$E_{baseline}$	Cost of Net Energy Consumption by baseline
$\$E_{DRL}$	Cost of Net Energy Consumption by DRL
$\%E_{saving}$	Energy savings %
abbreviation	Explanation for the abbreviation
$E_{baseline}$	Net energy consumption by baseline
E_{DRL}	Net energy consumption by DRL
E_{net}	Net energy consumption
E_{saving}	Energy savings
P_{batt}	Net battery power
P_{bld}	Building load
P_{pv}	PV generation
$\Delta\texttt{shed}_{hp}$	Differential shed
$\Delta\texttt{take}_{low+med\ p}$	Differential take
$\eta_{overshoot}$	Number of timestamps when ZAT overshot thermal comfort band
η_{total}	Total number of occupancy timestamps
μ_v	Mean of comfort violation
v	Comfort violation
σ_v	Standard deviation of comfort violation
ζ_v	Comfort violation in (°C-h)
a	Total floor area
$P_{baseline,hp}$	Baseline demand during high price
$P_{baseline,low+med\ p}$	Baseline demand during low+medium price
$P_{DRL,\ hp}$	DRL demand during high price
$P_{DRL,\ low+med\ p}$	DRL demand during low+medium price

that coordinate diverse end-use systems [12]. However, MPC requires a dynamic model (typically based on first principles or a mix of physics and data) of the system under control. Developing these models need significant expertise and it is time consuming, making this approach difficult to scale. Since buildings are heterogeneous, different models are required for each building. Without addressing these scalability issues, MPC algorithms for DERs cannot be broadly implemented.

With the development of machine learning algorithms and availability of inexpensive computing power, Deep Reinforcement Learning (DRL) has emerged as a promising alternative to MPC for managing DERs and building loads [13]. Hence, this work explores the development of a DRL controller and its deployment in an actual building to study the benefits and challenges of such approach and to evaluate potential scalability issues. While there are multiple ways to develop a DRL controller, the process generally involves formulating an agent that is in a partially observable environment and learning the best decisions through interactions with the environment. The agent observes environment snapshots, and chooses an action, receiving a reward value for this action in the current state. It continues to receive feedback on its actions until a terminal state is reached. The objective is to maximize the cumulative reward over all actions in the time the agent is active [14]. Nevertheless, allowing a DRL controller to train directly in a real building (the environment) by actively changing setpoints (*online* learning) could result in discomfort to the occupants and damage to the equipment. Hence, in the literature, the agent is sometimes trained using historical operational data before deploying it to real buildings (*offline* learning). However, as operational data from an existing building might be insufficient to represent all possible system states, oftentimes simulation models of the building (e.g., using simulation software such as MATLAB and EnergyPlus™) are used to train the DRL controller [15].

This paper proposes a DRL-based approach to control DER and test it in a real building that includes building PV and electric battery storage. The rest of the paper is organized as follows: in Section 2 we review the key concepts of Reinforcement Learning (RL) 2.1, present a summary of the existing research that have applied RL in building controls 2.2, and consequently, we identify the research gaps in the literature and state the contributions of this paper 2.3. In Section 3 we present the methodology, including the details about the field test and experimental setup, and in Section 4 we describe the design and training of the DRL algorithm. In Section 5 we present the results of the experiment, and Section 6 dives into detail about some of the interesting topics of discussion, including future directions of research. The paper is summarized and concluded in Section 7.

2. Related work

This section details the reinforcement learning background, specifically the Markov Decision Process. It also discusses the Deep Deterministic Policy Gradient (DDPG) method and how DRL can be used for building controls. Finally it highlights the research gaps that were identified through literature review and the contributions of this paper.

2.1. Learning via Markov decision processes

2.1.1. Reinforcement learning overview

Reinforcement learning (RL) is a class of machine learning algorithms that are based on a trial-and-error learning approach. An RL agent interacts with the environment, learns the dynamics by directly trying different control actions and observing the consequences through some notion of rewards. This typically involves the agent trying a significant number of actions (e.g., HVAC setpoints, battery setpoints) from a possible action space in the environment (e.g., building, DER system) that is in one of the many possible states (such as a specific time of the day when the building is occupied or outdoor temperature is above a certain limit) and receives a reward. The rewards indicate to the RL agent how well a particular action performed with respect to some environmental condition (e.g., thermal comfort, energy cost) [14].

Assuming that the environment is a fully observed collection of states, such that the observation at time t is equal to the environment state at time t (i.e., s_t), the sequential interaction between the RL agent and the environment can be modeled as a Markov Decision Process (MDP), which means the future state s_{t+1} of the environment is only dependent on the current state s_t (i.e., given the present state, a future state is independent of the past states) [16]. Formally, for an MDP the state transition probability is defined as:

$$\mathcal{P}(s_{t+1}|s_t, a_t, s_{t-1}, a_{t-1}, \ldots) = \mathcal{P}(s_{t+1}|s_t, a_t) \tag{1}$$

An MDP is defined as a four element tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$, where $\mathcal{S}$ is the set of states of the environment, $\mathcal{A}$ is the set of possible actions, $\mathcal{P}$ is the state transition probability that describes the probability distribution of next steps (s_{t+1}) given the current state (s_t) and action (a_t), $\mathcal{R}$ is the reward function that provides the reward obtained of taking action a_t at state s_t, and finally $\gamma \in [0, 1]$ is a parameter that is called *discount factor,* which determines the importance of future rewards. If $\gamma = 0$ the agent will be concerned only by maximizing immediate rewards, which means it will learn a control policy that selects actions to maximize R_{t+1}. As γ increases, the RL agent becomes increasingly focused on the future rewards. Given an MDP, the RL agent learns, by mapping the environment states to the actions, a control policy ($\pi(a_t|s_t) : S_t \rightarrow A_t$) that maximizes the expected cumulative reward at each time step (i.e., maximizing the expected cumulative reward it will receive in the future) [17].

Further, RL algorithms can be subcategorized into different groups: on-policy versus off-policy, online versus offline learning, and model-based versus model-free algorithms. Off-policy algorithms, as compared

to on-policy ones, have a greedy learning approach. In on-policy learning, the RL controller records rewards based on the current strategy/action performed, while in off-policy, the RL greedily selects the actions that gave the best performance from memory. Online learning means the RL agent is deployed in the real-world environment and RL agent is trained by interacting with the environment. Offline learning means the RL agent is trained without directly interacting with the environment. Instead, it learns from historical data or in a virtual training environment. In offline learning, the controller is deployed in the real environment once it is well trained. Given this nature, offline learning is "less risky" because the controller does not need to interact with the environment when it is not well trained. In model-based learning, the controller learns the system dynamics first and then uses the learned system dynamics for planning; while model-free algorithm learns the optimal control without learning the system dynamics. The model-based algorithm is usually more computationally expensive, because the algorithm first needs to learn an accurate environment model (usually a difficult task in real application), then it needs to find an optimal policy. Thus, model-free algorithms are more popular, as they are usually less computationally expensive.

Action value function The action-value function represents the expected cumulative reward of the RL agent starting from state s and following control policy π. Formally the action-value function is defined as:

$$Q_\pi(s,a) = \mathbb{E}_\pi\left[\sum_{k=0}^{T}\gamma^k r_{t+k+1}|S_t = s, A_t = a\right] \quad (2)$$

where $\mathbb{E}_\pi$ is the expected value given that the RL agent follows the control policy π. T is a final time step of the episode. Depending on the considered environment, the value of the discount factor γ need to be tuned based on the need to balance the current and future rewards.

Q-Learning The action-value function, which is also known as *Q-value,* gave its name to one of the major off-policy and model-free RL algorithms named *Q-learning.* The process of Q-learning algorithms is to compute and update at each iteration the Q-value for each state–action pair at time t, in order to achieve the maximum cumulative rewards. The optimal $Q^*_\pi(s_t,a_t)$ for action a_t taken at state s_t can be expressed using a recursive relation known as a *Bellman equation,* which is a summation of the present reward and the maximum discounted future rewards:

$$Q^*_\pi(s_t,a_t) = r(s_t,a_t) + \gamma max Q_\pi(s_{t+1},a_{t+1}) \quad (3)$$

When used in a discrete state–action space, the RL agent policy is determined by a state–action lookup table called a *Q-table,* which is used to select an action for a given state. The Q-table is updated according the following equation:

$$Q_\pi(s_t,a_t) \leftarrow Q_\pi(s_t,a_t) + \alpha\left[r(s_t,a_t) + \gamma max_{a'\in\mathcal{A}} Q_\pi(s_{t+1},a') - Q_\pi(s_t,a_t)\right] \quad (4)$$

where $\alpha \in [0,1]$ is the learning rate.

Deep Q-Learning The Q-table approach can work well for environments where the state–action space is relatively small, but when the states and action spaces increase or become infinite (i.e., environment with continuous states/actions), the size of the table becomes intractable. A solution to this problem is to replace the Q-table with a nonlinear representation (i.e., function approximation) that maps state and action onto a Q-value, which can be seen as a supervised learning problem (i.e., regression). Recently, approximating the action value function with a deep neural network has became one of the most popular options, due to network's capacity to accurately approximate high dimensional and complex systems. A simplified illustration of the Q-learning algorithm using an approximation function is provided by the following pseudo-code.

Several papers used neural network to approximate action-value functions [18,19]. However, these early methods often resulted in being unstable. This instability was due to: (1) the correlation between sequential observations, (2) the correlation between the action-values

Algorithm 1: Q-learning algorithm using approximation function

1. Initialize $Q(s_t,a_t)$ (i.e., initialize the neural networks with random weights).
2. Obtain the observation of the transition (s_t,a_t,r_t,s_{t+1}) by making the RL agent interact with the environment.
3. Compute the loss function:
$\mathcal{L} = \left(Q_\pi(s_t,a_t) - \left[r(s_t,a_t) + \gamma max_{a'\in\mathcal{A}} Q_\pi(s_{t+1},a')\right]\right)^2$.
4. Update Q (e.g., using the stochastic gradient descent algorithm) by minimizing $\mathcal{L}$ with respect of the neural network parameters.
5. Repeat starting from 2 until some convergence criteria is satisfied.

$(Q_\pi(s_t,a_t))$ and $r(s_t,a_t)+\gamma max_{a'} Q_\pi(s_{t+1},a')$, and (3) the fact that a small change to the action-value function may significantly change the policy.

In 2015, Mnih et al. [20] introduced Deep Q-Network (DQN), which was the first successful use of a combination of deep neural network and Q-learning algorithm. This work is responsible for the rapid growth of the field of deep reinforcement learning (DRL). In addition to the innovation of using neural networks, DQN proposed the use a replay buffer and two neural networks to approximate the action-value function and to overcome the instability issues of previous approaches. The replay buffer is used to store a relatively large number of observations of transitions (s_t,a_t,r_t,s_{t+1}). Mini-batches of these transitions are sampled randomly (using a uniform distribution), which allows the algorithm to update the neural network from a set of uncorrelated transitions at each iteration. To reduce the correlation between the action-values $(Q_\pi(s_t,a_t))$ and $r(s_t,a_t)+\gamma max_{a'} Q_\pi(s_{t+1},a')$, DQN uses two neural networks. The weights of the first one are updated at each iteration, and this network is directly interacting with the environment. The weights of the second neural network, called a *target network,* are updated after a fixed number of iterations by simply copying the weights of the first network.

While DQN can handle environments with high dimensional state spaces, it can only be used with discrete and low-dimensional action spaces. In fact, DQN relies on finding the action that maximizes the action-value function, which requires an iterative optimization at every step, if used with an environment that has continuous action. In theory, one can discretize the action space; however, this solution is likely be intractable for problems that require fine control of actions (i.e., finer grained discretization). The high number of discrete actions spaces are difficult to explore efficiently. Moreover, a naive discretization strategy of the action-spaces can exclude important information about the action domain, which can be essential for finding optimal control policy in several problems. To overcome this issue of discrete action spaces, [21] introduced the Deep Deterministic Policy Gradient (DDPG) algorithm, which is a model-free, off-policy algorithm that can learn control policies in high-dimensional, continuous action spaces.

2.1.2. Deep deterministic policy gradient

DDPG [21] is an actor–critic framework based on the deterministic policy gradient (DPG) method [22], and it borrows methodological advances (i.e., replay buffer and target networks) developed for the DQN algorithm [20]. An *actor–critic algorithm* is a RL algorithm that has two agents: an actor and a critic. The actor makes decisions based on observation of the environment and the current control policy. The role of the critic is to observe the state of the environment and the reward obtained by using an action and return the value-function estimate to the actor.

DDPG learns two networks that approximate the actor function $\mu(s_t|\theta^\mu)$ and the critic function $Q(s_t,a_t|\theta^Q)$, where s_t is the state of the environment at time t, a_t is the control action (i.e., the HVAC setpoints and the battery setpoint), and θ^μ and θ^Q are respectively the weights of the actor and critic networks.

The actor function $\mu(s_t|\theta^\mu)$ deterministically maps states to a specific action and represents the current policy. The critic function $Q(s_t, a_t|\theta^Q)$ (i.e., the action's values function) maps each action-state pair to a value in $\mathbb{R}$ that is the expected cumulative future reward obtained by taking action a_t at state s_t following the policy. During the training process the actor and critic networks are iteratively updated using stochastic gradient descent (SGD) on two different losses, L_μ and L_Q, which are computed using mini-batches of transitions samples (s_t, a_t, r_t, s_{t+1}). s_{t+1} is the resulting state by taking action a_t at state s_t, and r_t is the subsequent reward. Using SGD assumes that the transition samples in the mini-batches are independently and identically distributed, which may not be the case when the observations are generated sequentially. Similarly to DQN, DDPG address this issue by sampling uniformly the transitions from a relatively large fixed size replay buffer. This allows the algorithm to update the actor and critic network from a set of uncorrelated transitions, at each training iteration. The replay buffer is populated by sampling transitions from the environment using the current policy, and after the buffer is fully populated, the oldest transitions are replaced by the new ones. Another method that DDPG borrows from DQN to improve the stability of the training process is the use of two target networks: $\mu'(s_t|\theta^{\mu'})$ and $Q'(s_t, a_t|\theta^{Q'})$. Their weights $\theta^{\mu'}$ and $\theta^{Q'}$ are slowly tracking the learned weight θ^μ and θ^Q.

As for most DRL algorithms, DDPG needs to explore the state-space in order to avoid converging to a local minimum that produces non-optimal policies. Usually, this exploration is performed by adding noise sampled from a predefined random process N to the actions produced by the actor network:

$$a_t = \mu(s_t|\theta^\mu) + N \tag{5}$$

where N can be either a Gaussian process or, as proposed in [21], an Ornstein–Uhlenbeck (OU) process that generates temporally correlated exploration. This solution makes the changes in action noise less abrupt from one time step to another.

The pseudo-code of DDPG is given below. This algorithm is the DRL approach used in this work.

Algorithm 2: DDPG algorithm

Initialize $Q(s_t, a_t|\theta^Q)$ and $\mu(s_t|\theta^\mu)$ networks with random weights θ^Q and θ^μ.
Initialize target networks $\mu'(s_t|\theta^{\mu'})$ and $Q'(s_t, a_t|\theta^{Q'})$ with $\theta^{\mu'} \leftarrow \theta^\mu$ $\theta^{Q'} \leftarrow \theta^Q$.
Initialize replay buffer B.
for *episode* $= 1, ..., M$ **do**
  Initialize a OU process $\mathcal{N}_t$ for exploration
  Obtain initial observation of state s_1
  for $t = 1, ..., T$ **do**
    Obtain action $a_t = \mu(s_t|\theta^\mu) + \mathcal{N}_t$
    Execute action a_t and compute the reward r_t and observe the new state s_{t+1}
    Store transition (s_t, a_t, r_t, s_{t+1}) in replay buffer B
    Randomly sample a mini-batch of N transitions (s_i, a_i, r_i, s_{i+1}) from B
    Set $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'})$
    Update θ^Q by minimizing the loss:
    $L = \frac{1}{N}\sum_{i=1}^{N}(y_i - Q(s_i, a_i|\theta^Q))^2$
    Update θ^μ using the sampled policy gradient:

$$\nabla_{\theta^\mu} \approx \frac{1}{N}\sum_{i=1}^{N} \nabla_{\theta^a} Q(s, a|\theta^Q)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu} \mu(s|\theta^\mu)|s = s_i$$

    Update the target networks:

$$\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau\theta^{Q'}); \theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau\theta^{\mu'})$$

2.1.3. Hyperparameter tuning

As seen in Henderson et al. [23], the sensitivity of various DRL techniques can impact the reward scale, environment dynamics, and reproduciblity of the experiments. Finding benchmarks for a fair comparison is a challenge, but OpenAI stable baselines can help provide initial comparison across the gym environments chosen. Complex hyperparameters can affect how quickly the DRL solutions reach consensus or produce a robust solution that achieves the maximum reward. A popular technique to find optimal hyperparameters is the grid search as illustrated by Liang et al. [24].

2.2. DRL for building and DER control

The literature on advanced control algorithms for optimizing a building's energy operations is large, as mentioned in Section 1. Given the scope of this work, in this section we focus on reviewing the growing body of research using RL-based algorithms to control buildings and DERs [13,15,25,26]. Researchers used DRL controllers to achieve different objectives: authors of [27–36] aimed at improving thermal comfort of building occupants while minimizing energy consumption, whereas others [37–47] aimed at improving energy efficiency and reducing cost. Some recent work also has looked at using DRL for better load management through peak reduction, load shifting, and better scheduling [18,48,49]. The actions taken by DRL controllers include generating supervisory control setpoints for the HVAC systems (e.g., variable air volume [VAV] boxes, chiller plants, room thermostats), such as temperature [27,28,30–33,37–39,41,42,49], flow rate [29,31,42,49] and fan speed setpoints [45]. Supervisory control of electrical batteries by setting the charge/discharge rate [35,44,46,47] is also common. This approach has the advantage of leaving the lower-level control of the equipment to local controllers that ensure quick response and equipment safety. For simpler systems, such as residential appliances, authors of [18,35,43,46,48] have also generated actions to turn them on and off directly. With a focus more on the supply side of the grid, Hua et al. also developed an asynchronous advantage actor–critic (A3C) algorithm based DRL controller to manage grid level renewable energy sources (subgrid level PV, Wind Turbine Generator) along with the building load, and trained the algorithm in simulation [50].

The reward functions used in the literature typically track the objective of the DRL algorithm, hence penalizing occupant discomfort and rewarding for energy savings (and corresponding cost savings) [18,27–35,37–43,45–49]. Control of on-site energy storage often includes an additional penalty to discourage battery depreciation [18,44,47].

Researchers have developed their DRL algorithms using a wide variety of algorithms: Actor–Critic [27,35], Asynchronous Actor–Critic [37,39–41], Policy Gradient [18,28], Deep Deterministic Policy Gradient [32,42,47,49], Trust Region Policy Optimization [38], Double Deep Neural Fitted Q Iteration [34], Q-learning [45,46,48], Deep Q-learning [18,29–31,44,49], and Proximal Policy Optimization [33].

Python (using the OpenAI Gym framework) and MATLAB have been the most popular programming environments used to develop these DRL controllers. Due to the difficulty and cost of field testing, a large portion of the existing work has been performed in simulation. In the realm of simulation-based research, DRL-based controllers have been developed for residential buildings [18,34,35,47,48], commercial buildings (e.g., single-zone, multi-zone, multi-storey) [27–29,31–33,39,49], and data centers [38,42]. In the few instances of field tests, DRL controllers have been deployed in well instrumented facilities, such as the Intelligent Workspace at Carnegie Mellon University, Pittsburgh, United States [36,37,40,41] with unconventional HVAC systems, or in simpler residential buildings [43,46].

Fig. 1. (a) Outside view of FLEXLAB, (b) mechanical drawing of FLEXLAB envelope and HVAC (top view) [51].

2.3. Research gaps and contribution

The literature clearly shows that DRL research in the built environment has great potential, and it is a fast growing field [13,15,25,26]. However, a clear gap identified in the literature is that most of DRL-based building energy optimization methods are still not implemented in practice [52] nor use physical equipment [13]. This observation is also echoed by various reviews such as [25]: "[...] The vast majority of studies that implement RL for problems relating to building energy management are in simulation only" and [26]: "[...] the research works about applications of RL in sustainable energy and electric systems are largely still in the laboratory research stage, which have not been practically implemented in sustainable energy and electric systems". A second gap is that very few RL studies have co-optimized DERs with HVAC control to achieve load flexibility at the building level. The few field tests that employ DRL-based algorithms only optimize the operation of one system at a time (mostly the HVAC system).

Given these gaps in the literature, the goal of this paper is to implement a price-responsive DRL-based controller and deploy it in an actual building that includes on-site distributed energy resources (DERs). The DRL algorithm will aim at minimizing energy cost while maintaining acceptable indoor thermal comfort. The performance of the DRL-based controller will be tested against a best-in-class rule-based controller. The paper makes the following contributions to the existing literature:

- An application of a DRL algorithm for integrated optimal control of a DER system composed of PV, electric storage, and thermal loads in a commercial building
- A field demonstration of the DRL controller in a well instrumented single-zone commercial building test facility, controlling multiple pieces of equipment, along with on-site DERs
- An open-source OpenAI Gym environment that can be reused by other researchers, in order to both train and evaluate the performance of their algorithms
- A discussion about the challenges and lessons learned in the field deployment of the DRL controller in an actual building, particularly with respect to integrating multiple building systems

3. Methodology

This section discusses the experimental design and test set up. It also outlines the performance metrics used to monitor and evaluate the outputs of different tests and assess the efficacy of the DRL controller. Finally, it discusses the baseline strategy used to control the battery and the building loads during the tests.

3.1. Experimental design and setup

The evaluation of the controller developed in this work was conducted in FLEXLAB [51], a well-instrumented experimental test facility at Lawrence Berkeley National Laboratory in Berkeley, California, United States. FLEXLAB allows accurate measurement of the behavior of integrated systems (e.g., HVAC and batteries) in providing grid services (e.g., Shift/Shed load when requested) and building services (e.g., visual and thermal comfort). To evaluate the performance of the DRL controller compared to a baseline, we used a pair of side-by-side identical cells (called Cell A and B in the paper), a unique feature of FLEXLAB (Fig. 1). Each cell is representative of a small office space with a floor area of 57 m^2 and includes a large south-facing window. FLEXLAB can be reconfigured with different types of equipment, end-uses, and envelope features. As HVAC we selected a single-zone variable-capacity air-handling unit (AHU) for each cell, a very common system installed in many U.S. buildings [53]. The AHU had a single variable-speed fan and two dampers for outdoor air and return air mixing. The air side of the AHU was monitored with sensors to measure air volume flow rate as well as the temperatures and humidities in each section of the ducts. The water-side of the two AHUs was served by a small chiller and boiler, shared by the two cells (Fig. 1). The flow rates and inlet and outlet temperatures in each AHU coil were measured to calculate heat flows transferred to the thermal zones. Power consumption of the heating and cooling equipment was estimated using these measured heat flows multiplied by a fixed efficiency (chiller COP = 3.0 and boiler efficiency = 0.95). Each room had six ceiling-mounted light fixtures, six plug-load stations (i.e., desks with desktop computers), six heat-generating mannequins that emulate occupants, and multiple environmental sensors measuring temperature, indoor light levels, and relative humidity (Fig. 2). The DERs comprised a 3.64 kilowatt (kW) photovoltaic system and a Tesla Powerwall battery storage with a capacity of 7.2 kWh and peak power output of 3.3 kW (Fig. 3) for each cell. Each sensor in FLEXLAB collected data every 10 s or less and it was aggregated to 1 min for analysis. The DRL controller was deployed in Cell B, while the baseline controller was deployed in Cell A. The FLEXLAB control system allowed users to change the setpoint of the supply air temperature ($\mathtt{SAT}_{\mathtt{sp}}$) and the supply air flow rate ($\mathtt{SAF}_{\mathtt{sp}}$) of the AHU directly or to influence them by setting the zone air temperature heating and cooling setpoints ($\mathtt{ZAT}_{\mathtt{hsp}}$, $\mathtt{ZAT}_{\mathtt{csp}}$) using a standard control sequence. In addition, the charge/discharge setpoint of the battery was controllable ($\mathtt{battery}_{\mathtt{sp}}$).

Three different scenarios were tested, based on demand-side management strategies commonly referred as: (1) Energy Efficiency (EE), (2) Shed, and (3) Shift [3]. In the EE scenario, the controllers were compared based on the amount of energy purchased from the grid; in the Shed scenario, the comparison was based on the ability to reduce energy purchases in a specific time window; and in the Shift scenario,

Fig. 2. Room set up inside FLEXLAB: heat-emitting mannequins or equivalent thermal generators, desktop computers, and overhead light-emitting diode (LED) lights with multiple environmental sensors [51].

the two controllers were evaluated based on their ability to shift energy purchases from one time window to another one. Different electricity prices were set to produce a desired effect on the electricity purchase (i.e., higher prices would drive reduced purchase from the grid).

In the first experiment, which ran for seven days in July, the electricity price was kept constant (0.13$/kWh). The Shed experiment involved two tiers of prices: a low price (0.13$/kWh) throughout the day, and a high price between 4 pm and 9 pm (1.33$/kWh). The Shift experiment included an additional price tier: high price (4 pm–9 pm) at 0.16$/kWh; medium price (2 pm–4 pm and 9 pm–11 pm) at 0.13$/kWh; and low price (11 pm–2 pm) at 0.11$/kWh. These prices were selected based on current commercial tariffs used in California [54] at the time of the experiment. New research on demand response [55,56] have suggested using prices for reflecting grid requirements and for initiating responses from building equipment, instead of using traditional demand-response events. The constant prices that were used in the EE experiment, along with the objective to minimize total energy costs, discourage any shifting of loads from one period to another and incentivize an overall reduction in energy consumption by optimizing the operations of the building equipment. The tiered prices used in the Shed and Shift experiments were used to trigger load reduction during the high price times and in the case of Shift, to encourage a corresponding additional increase in load during the lower price periods as well, thereby achieving a load shed and load shift.

For simplicity, the tariff only accounted for variable cost of electricity (i.e., kWh) and not for maximum demand (i.e., kW). Both the baseline and the DRL algorithm were tasked to minimize the energy cost while maintaining a zone air temperature deadband of 21 °C–24 °C during the HVAC operational hours (7 am–7 pm) and 15 °C–29 °C during the remaining hours of the day. The experiments were conducted during the cooling season, at the end of July and early August of 2020, with each experiment spanning 7–8 days, with a total testing period of slightly more than three weeks.

3.2. Performance metrics

In each scenario, the performance of the controllers was evaluated using the following metrics:

1. ENERGY CONSUMPTION
   Since the DER system is composed of local generation and storage, the energy consumption is defined in terms of energy purchase from the grid, which correspond to the difference between the PV generation and the building load adjusted by the change in state of charge of the battery. The building load is a summation of power consumption by HVAC systems (chiller, fans, pumps), lighting, and plug loads. We call this *net load*.

$$E_{net} = \int_0^T (P_{bldg} + P_{batt} - P_{pv})\, dt \tag{6}$$

   (a) ENERGY SAVINGS
   The energy savings are the difference between net energy consumption in Cell A (baseline) and Cell B (DRL). They are expressed in kWh for the whole testing period of each scenario:

$$E_{saving} = E_{baseline} - E_{DRL} \tag{7}$$

   (b) % ENERGY SAVINGS
   The % savings correspond to the energy savings divided by the baseline:

$$\%\, E_{saving} = \frac{E_{baseline} - E_{DRL}}{E_{baseline}} \times 100\% \tag{8}$$

2. ENERGY COST
   The energy cost is the product of E_{net} and energy price, that differ by scenario and time period.

   (a) COST SAVINGS
   Similar to energy savings, cost savings are defined as the difference between cost in Cell A (baseline) and Cell B (DRL). They are expressed in $ for the whole testing period of each scenario.

   (b) % COST SAVINGS
   The % COST corresponds to the cost savings divided by the baseline:

$$\$_{savings}\% = \frac{\$(E_{baseline}) - \$(E_{DRL})}{\$(E_{baseline})} \times 100\% \tag{9}$$

(a) 3.64 kW local solar power generation

(b) 7.2 kWh local energy storage capacity with a peak output of 3.3 kW (only two of the three shown have been used in this experiment)

Fig. 3. Distributed Energy Resources installed in FLEXLAB [51].

3. THERMAL COMFORT

Thermal comfort is measured in terms of violations of the indoor or zone air temperature (ZAT) from the comfort band during building occupancy (between 8am – 6pm). The thermal comfort range is defined as 21°C – 24°C.

(a) % TIME OUTSIDE COMFORT BAND

This metric captures the relative duration of violations compared to the testing period.

$$\eta_{\text{overshoot}}\% = \frac{\eta_{\text{overshoot}}}{\eta_{\text{total}}} \times 100\% \tag{10}$$

(b) MEAN AND STANDARD DEVIATION OUTSIDE COMFORT BAND

The mean and standard deviation of the violation of comfort measure the intensity and variability of such events.

$$\mu_v = \sum_{0}^{\eta_{total}} \frac{\max(0, \text{ZAT} - 24\ °\text{C}) + \max(0, 21\ °\text{C} - \text{ZAT})}{\eta_{\text{total}}} \tag{11}$$

$$\sigma_v = \sqrt{\frac{\sum([\max(0, \text{ZAT} - 24\ °\text{C}) + \max(0, 21\ °\text{C} - \text{ZAT})] - \mu)^2}{\eta_{\text{total}}}} \tag{12}$$

(c) DEGREE HOUR OUTSIDE COMFORT BAND

Degree hour is a metric used to evaluate the performance of the controllers. It is defined as the integral of ZAT overshoot beyond 24°C over the duration of the overshoot (T) and undershoot below 21°C. The factor of 0.25 in Eq. (13) represents 15 minute time interval of each timestamp.

$$\zeta_v = \sum_{0}^{\eta_{total}} (\max(0, \text{ZAT} - 24\ °\text{C}) + \max(0, 21\ °\text{C} - \text{ZAT})) \times 0.25\text{t} \tag{13}$$

where t = number of timestamps when ZAT is outside the comfort band

$$\zeta_v \text{ per day} = \frac{\zeta_v}{\#\text{ test days}} \tag{14}$$

4. SHED and SHIFT METRICS

The differential load shed (W/ft2) during peak price time is defined as the difference in the average demand intensity between the baseline and DRL controller during the high price period [57].

$$\Delta\text{shed}_{\text{hp}} = \frac{\text{P}_{\text{baseline,hp}} - \text{P}_{\text{DRL,hp}}}{\text{a}} \tag{15}$$

Demand shifted during the test can be evaluated using two metrics: a demand reduction metric during the high price period (identical to the demand shed metric) and a demand increase metric during the rest of the day (take period). Demand increase is considered negative and demand decrease is considered positive, as per conventions from traditional demand response programs.

$$\Delta\text{take}_{\text{low+med p}} = \frac{\text{P}_{\text{baseline,low+med p}} - \text{P}_{\text{DRL,low+med p}}}{\text{a}} \tag{16}$$

3.3. Baseline control strategy

Lighting, plugs, and occupants (i.e., mannequins) were set on a time-based schedule and operated identically in the two cells. The AHU in the baseline cell implemented Section 5.18 of ASHRAE Guideline 36 (Single Zone VAV Air-Handling Unit) [58]. However, a supervisory rule-based controller that generated the zone temperature setpoints (ZAT_{hsp}, ZAT_{csp}) for the AHU was used in the shift and shed experiments to implement "pre-cooling" (for an appropriate comparison to the advanced DRL controller). A similar rule-based battery charge/discharge setpoint ($\text{battery}_{\text{sp}}$) generator was also implemented for all three experiments. The rules for both these controllers are tabulated in Tables 1, 2, and 3. Similar to the DRL controller, these rule-based supervisory baseline controllers generated new setpoints for the AHU and the battery every 15 min.

Table 1
Baseline control strategies for the EE test.

Equipment	Control/Parameters	Time
HVAC	ZAT_{hsp} = 21 °C; ZAT_{csp} = 24 °C	7 am–7 pm
	ZAT_{hsp} = 15 °C; ZAT_{csp} = 29 °C	All other times

Battery	if(P_{pv}>P_{bldg}):
	charge battery (max: 90%)
	else:
	discharge battery (min: 10%)	

Table 2
Baseline control strategies for the SHED test.

Equipment	Control/Parameters	Time
HVAC	ZAT_{hsp} = 21 °C; ZAT_{csp} = 24 °C	7 am–4 pm
	ZAT_{hsp} = 15 °C; ZAT_{csp} = 26 °C (load Shed during high prices)	4 pm–7 pm
	ZAT_{hsp} = 15 °C; ZAT_{csp} = 29 °C	All other times
Battery	charge battery (max: 90%)	7 am–4 pm
	discharge battery (min: 10%)	4 pm–7 pm

Table 3
Baseline control strategies for the Shift test.

Equipment	Control/Parameters	Time
HVAC	ZAT_{hsp} = 21 °C; ZAT_{csp} = 24 °C	7 am–7 pm
	ZAT_{hsp} = 17 °C; ZAT_{csp} = 20 °C (pre-cooling during low prices)	12 pm–2 pm
	ZAT_{hsp} = 16 °C; ZAT_{csp} = 22 °C (pre-cooling during medium prices)	2 pm–4 pm
	ZAT_{hsp} = 15 °C; ZAT_{csp} = 26 °C (load shed during high prices)	4 pm–7 pm
	ZAT_{hsp} = 15 °C; ZAT_{csp} = 29 °C	All other times
Battery	charge battery (max: 90%)	7 pm–2 pm
	if (P_{pv}<P_{bldg}): discharge battery (min: 50%)	2 pm–4 pm
	discharge battery (min: 10%)	4 pm–7 pm

It should be noted that the equipment in the baseline-controlled cell was programmed with a preheating mode that did override these supervisory setpoints. This is a common strategy that building managers use in many building systems and it was implemented by default in the test cells. While the test was conducted in summer, heating triggered during several days, because in the local climate (Berkeley, CA) it is not uncommon to have large diurnal temperature swings with outdoor temperature dropping to around 12°C at night. This resulted in heating events in the early morning hours during the actual experiments.

4. DRL control algorithm

The DRL control algorithm developed uses a model-free learning approach, namely an actor–critic technique called DDPG, which is suitable for continuous control problems. While the controller does not need a model, it needs data to be trained on. Sometimes, building operational data from an existing building is insufficient to represent all possible system states. In our case, the historical building operational data available was generated using a single control logic and did not explore all possible control actions. In contrast, since DRL needs a trial-and-error approach, i.e. testing different control logic, to figure out which one performs the best, a building simulation model was used

to train the DRL controller. A training framework that uses a physics-based simulation component was implemented to learn end-to-end optimal control policy to manage the charge and discharge of battery storage and HVAC setpoints. We calibrated the simulation model using operational data from the testbed, and we are confident that this model can emulate the real environment (FLEXLAB) reasonably well.

4.1. Simulation environment

4.1.1. Software stack and modeling choices

The EnergyPlus™ simulation tool [59] was used to model the building and HVAC system. The physical parameters of the FLEXLAB envelope were selected based on the materials in the construction drawings. For the HVAC system, we developed a water-based primary loop and air-based secondary loop HVAC model, as shown in Fig. 4, which represents the FLEXLAB configuration used for the test. We assumed constant boiler efficiency (0.95) and chiller COP (3.0) because climate at the test site is mild.

The battery and PV panels were simulated using the Smart Control of Distributed Energy Resources (SCooDER) library [60]. SCooDER is a Modelica library developed to facilitate the simulation and optimization of photovoltaics, battery storage, smart inverters, electric vehicles, and electric power systems [60]. SCooDER depends on the Modelica Standard Library and Modelica Buildings Library. The interaction between the DRL controller and the four simulated components (i.e., building envelope, HVAC system, battery, PV generation) is a typical co-simulation problem, since multiple software packages are involved. After a review of existing simulation frameworks, we selected the Functional Mock-up Interface (FMI) [59], a standard that defines a container and an interface to exchange dynamic models using a combination of XML files, binaries, and C code zipped into a single file [59]. As an open-source standard, the FMI is used in both academia and industry, and currently supported by more than 100 tools. By using the FMI, the building envelope, PV, and battery model were all exported in accordance to the FMI standard as Functional Mock-up Unit (FMU). The DRL controller was implemented in a Python environment using the package PyFMI [61] that enabled the interaction with the exported FMU. The FMUs for the different models were further wrapped into an OpenAI Gym environment, to run the building simulation in parallel with the DRL algorithm, developed in PyTorch.[2] To allow others to reproduce the analysis or test other DRL algorithms, the simulation and control framework were packaged into a Docker container including all the dependencies needed. The software is distributed as an open-source library and can be found at https://github.com/LBNL-ETA/FlexDRL.

4.1.2. Model calibration

The simulation model was calibrated with experimental data gathered during a calibration test performed before the experiment. We first calibrated the envelope model by comparing the indoor temperatures of the physical building to the indoor temperatures reported by the simulated model. To accurately measure the stratification and nonuniform distribution of temperatures, we deployed four sensor trees with seven temperature sensors in Cell A and Cell B. In a detailed physics-based model with hundreds of parameters, such as the one developed in EnergyPlus used in this paper, there are different combinations of variables that produce the same results, therefore their calibration is challenging. To select the best parameters needed for calibration we followed best practices in the literature. Zhang et al. (2019) [62] used the sensitivity analysis proposed by Morris (1991), [63] to identify four key parameters, which are found to significantly influence the simulation errors and need to be calibrated. The identified parameters include the (1) thermal insulation, (2) total area of radiant heating/cooling surfaces, (3) internal thermal mass multiplier, which is defined as the ratio of total interior thermal mass to the thermal capacitance of the air in the volume of the specified zone, and (4) infiltration rate, i.e. air changes per hour. We tuned the three parameters that are found to be the most effective in Zhang et al. (2019) [62] and relevant to our experiment settings: thermal insulation, internal thermal mass multiplier and infiltration rate to minimize the difference between the measured and simulated temperature. We measured the indoor temperature at 1-minute intervals and up-sampled it to 15 min, to match the simulation time step. We calculated the Coefficient of Variation of Root Mean Square Error (CV-RMSE) between the 15-min interval measured and simulated temperature. By adjusting the internal thermal mass multiplier and infiltration rate, we achieved a CV-RMSE of less than 3% with the internal thermal mass multiplier of 6 and the infiltration rate of 2. We summarized the improvements of CV-RMSE during the model calibration process in Table 4.

Table 4
Summary of calibration measures and CV-RMSE.

Cell	Calibration measures	CV-RMSE
Cell A	Raw model, developed from design document	11.2%
	Tuning thermal insulation	7.6%
	Tuning thermal insulation & infiltration rate	4.0%
	Tuning thermal insulation infiltration rate & thermal mass multiplier	2.8%
Cell B	Raw model, developed from design document	10.9%
	Tuning thermal insulation	6.3%
	Tuning thermal insulation & infiltration rate	2.9%
	Tuning thermal insulation infiltration rate & thermal mass multiplier	2.1%

In addition, we calibrated the power consumption of the fan, which is another major energy consumer in air-based systems. We used a cubic polynomial curve to model the fan energy behavior and achieved a CV-RMSE of 7% for the fan in Cell A and 8% for fan in Cell B when comparing the air flow values recorded at 15-min intervals. Then we used the regressed coefficients to develop the fan model. Since the chiller and boiler serve both the baseline and the DRL cell, and they use a secondary loop with storage, it was impossible to calibrate the power curve of each cell independently, and default values were used. In this study, we set COP to 2.5.

4.2. Design of the DRL algorithm

4.2.1. State and actions

The variables of the state vector s_t were selected as: time of the day, outdoor temperature, outdoor relative humidity, solar irradiation, zone temperature, net power, net energy, battery state of charge, PV generation power, and electricity price look ahead. The DRL controller ran every 15 min and generated three actions (i.e., three setpoints): supply air temperature of the AHU; supply air flow rate of the AHU; and charge/discharge rate of the battery. These actions were mapped to model variables in the simulation, as well as to actual controllable points in FLEXLAB (respectively $\texttt{SAT}_{\texttt{sp}}$, $\texttt{SAF}_{\texttt{sp}}$, and $\texttt{battery}_{\texttt{sp}}$).

4.2.2. Reward function

The goal of DRL control algorithm was to find an optimal policy that minimizes the cost of the electricity purchased from the grid while maintaining a good level of thermal comfort. In other words, the objective was to efficiently consume the electricity locally produced by the PV panels and manage the battery while maintaining the indoor zone temperature within a desired range. The considered reward function was composed of three parts: (1) the penalty due to the energy consumption E_{cost} (replaced by energy cost in the Shed and Shift scenario), (2) the penalty due to the violation of the temperature comfort zone C, and (3) the penalty due to the violation of the battery physical limits λ. This reward function is defined as:

$$r_t = -\alpha E_{cost}(a_t, s_{t+1}) - \beta C(a_t, s_{t+1}) - \lambda(a_t, s_{t+1}) \quad (17)$$

[2] https://pytorch.org/.

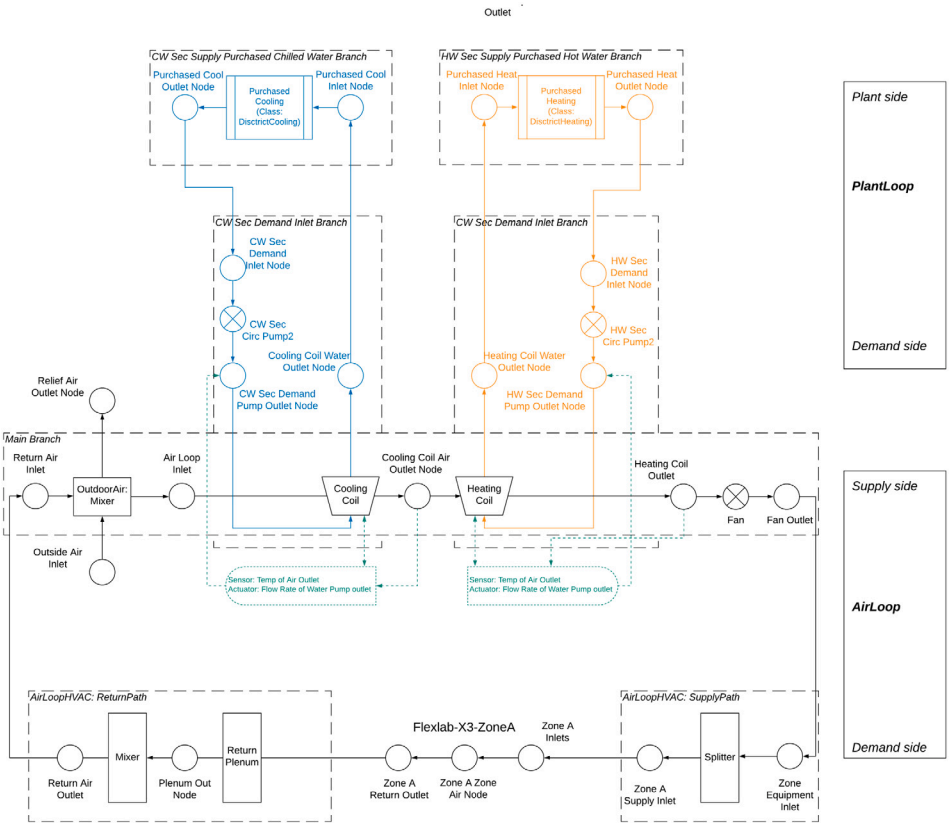


Fig. 4. HVAC system schematics in the test facility and Energyplus model.

with

$$C(a_t, s_{t+1}) = 1 - exp(-0.5(T_{t+1} - T_m)^2) + 0.2([T_{t+1} - T_U]_+ + [T_L - T_{t+1}]_+) \quad (18)$$

and

$$\lambda(a_t, s_{t+1}) = \begin{cases} -20, & \text{if } E_b(t+1) < 0 \text{ and } |E_b(t+1)| > SOC_{min} \\ -20, & \text{if } E_b(t+1) > 0 \text{ and } (E_b(t+1) + SOC(t)) > SOC_{max} \\ 0, & \text{otherwise.} \end{cases} \quad (19)$$

where T_{t+1} is the zone temperature at time $t+1$ (i.e., after action a_t was applied), T_m is the average desired temperature, and T_L and T_U are (respectively) the desired lower and upper bound for the zone temperature. $E_b(t+1)$ is the total energy that the battery has been charged ($E_b(t+1) > 0$) or discharged ($E_b(t+1) < 0$) between t and $t+1$. $SOC(t)$ is the state of charge (SOC) of the battery at t, SOC_{min}, and SOC_{max} are respectively the minimum and the maximum battery state of charge that is allowed. To ensure that the equation is homogeneous, the units of α, β and λ are set accordingly.

Using an approach introduced in [38], the penalty due to the violation of the temperature comfort zone C was defined as shown in Fig. 5. Within the red bars (temperature comfort boundaries), the reward is shaped like a bell curve that has a maximum reward (i.e., equal to 0) when T_t is equal to T_m, and rapidly decreases as the temperature moves away from T_m. Outside the red band we imposed a linear decay (i.e., trapezoid shape) of the reward. We found that adding the bell curve component to the thermal penalty, as shown in Fig. 5, significantly improved the stability of the control algorithm, especially when the cost of energy was equal to 0 (i.e., during the periods when all the load was covered by the PV generation).

The reward component that penalized the violation of the battery physical limits $\lambda(a_t, s_{t+1})$ was designed to keep the battery from making charging/discharging decisions that violate the system's physical constraints. This is achieved by adding a high penalty on actions that violate the constraints. This approach allows the actor to easily limit the space of actions where the optimal value is searched. However, our tests showed that it is important to select the proper penalty value (i.e., −20 in this work) because if the penalty is too high the actor may avoid any control action on the battery (i.e., the controller converged to a local maximum), because they consider these actions too risky (i.e., the cumulative reward can be too negative).

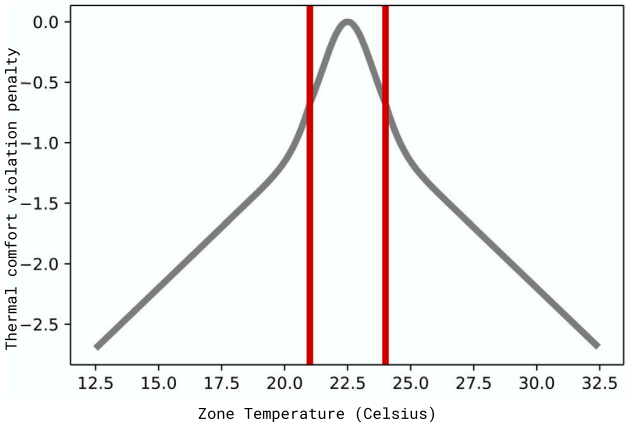


Fig. 5. Thermal comfort penalty function. The vertical axis correspond to the penalty due to the violation of the temperature comfort zone C and the horizontal axis correspond to the zone temperature in degree Celsius.

4.2.3. DRL algorithm hyperparameter tuning

The simulation framework developed was used to tune several hyperparameters of the DRL controller, to achieve a good control policy:

- **mini-batch size**: number of transitions (i.e., (s_t, a_t, r_t, s_{t+1})) samples used by the gradient-based optimizer.
- **actor and critic learning rate**: optimization parameter that controls how strongly the network weights are updated while moving toward the minimum of the loss functions.
- **neurons number**: the size of the two layers that constitute each network.
- **discount factor** γ: parameter used in critic loss function to control up to what extent future rewards have an impact on the return at time t. In other words, how many future time steps the agent is considering to select an action.
- **OU parameters**: θ the reversion rate and σ the standard deviation, which are two parameters of the OU process that will control level of the state space exploration.
- **reward function weights**: weights that control the importance of electricity cost vs. the thermal comfort.
- **electricity price look ahead**: the number of following hours of electricity price.

The performance of the DRL policy depends significantly on these parameters. A manual selection approach is extremely time consuming, requires significant DRL expertise, and the interaction of the hyper-parameters can be counterintuitive. Thus, we adopted a random search grid [64] approach to tune the parameters. This method randomly selects combinations of hyper-parameters from a uniformly distributed search space. To limit the computational cost, we fixed the number of combinations selected to 25 for each test batch. The hyper-parameters that obtained the highest reward, estimated using the data from the test year (see next section), were selected as the best parameters. A common problem with DRL approaches is the high variability of the resulting policies when different random seeds are used for the same combination of hyper-parameters. To deal with this challenge, seven additional policies using a different random seed were trained for each best tuned combination (one for each test batch). The policies that were deployed in FLEXLAB were selected from among those eight policies as the ones that produced the highest reward using the test year. Note that a more robust approach would have been to include the random seeds as hyper-parameters to be tuned and to test all the considered seeds for each hyper-parameter combination. However, this approach is very costly in term of computational complexity, therefore it was not pursued in this work.

4.2.4. DRL controller training

Given the availability of five years of local weather data, four years were used to train the DRL controller, using the simulation framework described in Section 4, and one year was used for validation. For each test scenario (i.e., EE, Shed, and Shift) we trained DRL models separately using the corresponding price signal. Since the state vector consisted of various physical features that have different units with significant differences in value ranges, we scaled the state observations to the range [−1,1]. Adam was used as a gradient-based optimizer [65]. The replay buffer size was fixed to 1.5 million transitions. Both actor and critic networks had the same size and had two layers. The energy cost reward function weight α was fixed to 1. For all three scenarios the selected hyperparameters were: batch size = 128; weight of thermal comfort β = 3; actor learning rate = 0.00005; critic learning rate = 0.0001; discount factor γ = 0.98; and OU σ parameter = 0.3. Additionally, the first and second network layer sizes were respectively: 450 and 400 for EE; 400 and 400 for Shed; and 400 and 350 for Shift. The OU θ parameter was 0.2 for EE and Shed, and 0.15 for Shift. The electricity price look ahead was two hours for Shift and three hours for Shed. Note that we defined each episode as one calendar year, thus one episode contains 35,040 time steps. Lawrencium[3] computational cluster resources have been used for training the controllers, and more specifically we have used nodes with INTEL XEON Gold 5218 (i.e., 32 cores central processing unit). Each training experiment (i.e., specific hyperparameters set) took about 3 days to converge.

Table 5
Metrics: DRL vs. baseline controller.

Test	Metric	Value
EE	% Energy savings	0.6%
	% Cost savings	0.6%
Shed	Differential shed during high price	0.25 W/sq.ft. (50%)
	% Cost savings	29.4%
Shift	Differential shed during high price	−0.34 W/sq.ft. (−97%)
	Differential take during low + medium price	1.1 W/sq.ft. (49%)
	% Cost savings	39.6%

Table 6
ZAT Overshoot % and parameters for DRL vs. Baseline controlled cells.

Controller	% of time intervals outside comfort band and violation rate	During the overshoot period (above 24 °C)		
		μ_v (°C)	σ_v (°C)	ζ_v per day (°C-h/day)
EE baseline	4	0.3	0.21	0.33
EE DRL	13	0.3	0.20	0.99
Shed baseline	25	1.0	0.98	4.92
Shed DRL	15	1.3	0.81	3.68
Shift baseline	15	0.7	0.53	2.48
Shift DRL	14	0.3	0.29	1.27

5. Results

The performance of the controllers was evaluated using the following metrics: reduction in cost (all tests), differential load shed during peak price time (shed and shift tests) or differential load take during low and medium price periods (shift test), and thermal comfort violations (all tests). Equations for these metrics are detailed in Section 3.2. The DRL algorithm reduced energy costs across all the three tests as compared to the baseline control strategy and the results are summarized in Table 5. These costs are based on the tariffs described in Section 3.1. In the Shed test, the DRL shed 0.25 W/sq.ft. (50%) more than the baseline controller during the high price period. However, during the Shift test, the DRL algorithm shed 0.34 W/sq.ft. (97%) less than the baseline strategy during the high price time and drew more power, 1.1 W/sq.ft. (49%) more than the baseline during the remaining day, while still reducing cost. This behavior was contrary to what we expected and it is discussed in detail in Section 6.3.

In terms of thermal comfort, neither the DRL nor the baseline control strategy were able to maintain thermal comfort in the cells at all times. Due to high internal and external loads and limited HVAC capacity, the temperature exceeded the upper threshold (i.e., 24°C) several times during each test. During the EE test, the overshoot during the occupied hours (8 am–6 pm) averaged over the test days was higher for the DRL (0.99°C-h) than the baseline (0.33°C-h), thus pointing to a poorer performance by the DRL controller. However, for both Shift and Shed tests, the DRL controller performed significantly better than the baseline controller, as summarized in Table 6.

5.1. Energy efficiency test

Fig. 6 shows the details about the EE test. As the cost of energy is fixed throughout the day, the expectation on DRL during the EE test is

[3] http://scs.lbl.gov/home.

Table 7
SHED TEST comparison of the DRL vs. baseline controllers' demand and total cost of energy consumed during different price periods.

Metric	Baseline	DRL
Mean demand in low price	594 W	655 W
Mean demand in high price	242 W	121 W
Mean daily energy consumption	12.5 kWh	13.1 kWh
Cost of energy consumed	$ 3.13	$ 2.46

Table 8
Shift test comparison of DRL vs. Baseline controllers' load demand and total cost during different price periods.

Metric	Baseline	DRL
Mean of demand in low price	870 W	412 W
Mean of demand in medium price	207 W	129 W
Mean of demand in high price	170 W	334 W
Mean of daily energy use across all test days	14.21 kWh	8.21 kWh
Cost of energy consumed	$ 1.62	$ 1.03

to minimize the energy consumption. In our experimental set up, this would mean that the DRL-controlled cell should either consume less energy than the baseline cell while providing similar thermal comfort or consume a similar level of energy while providing better thermal comfort. While the aggregated energy consumption and cost of the two controllers was similar, the baseline controller used more energy for heating (Fig. 6c), while the DRL controller used more energy for cooling, especially during the days with mild outdoor temperatures (Fig. 6b). This also impacted the internal temperature profile (Fig. 6d). The baseline controller kept consistently lower temperatures in the test cell, but that strategy caused more heating in the morning, after the room was unoccupied and exposed to relatively cold outdoor temperatures at night (between 13°C and 19°C). The temperature violations, all above the comfort band, were slightly higher in the DRL cell (Table 6). The DRL controller seemed unable to reduce the indoor temperature fast enough at the peak of internal and external heat gains. The batteries were not significantly utilized by either controllers, remaining close to their minimum charging level for most of the time (Fig. 6a). Since the cost of electricity was constant in the EE test, the energy and cost performance were aligned.

5.2. Shed test

The objectives of the DRL controller are to minimize both energy costs and the thermal discomfort. The anticipated behavior of the DRL controller during the high price period (4pm–9pm) of the Shed test was to reduce consumption by relaxing the comfort band. While there was no expectation to consume more energy any other time, we also expected the controller to pre-cool the zone before this period, since this strategy would allow the HVAC system to idle during the high price period, leading to cost savings and a reduction in thermal discomfort. Additionally, we expected the battery to be charged as much as possible before the high price period to be available for discharge when the cost of electricity is higher. Fig. 7 shows that DRL reduced cost significantly during the Shed test, while it reduced energy consumption only marginally. As noted earlier in Table 5, the differential load shed during the high price time is 0.25 W/sq.ft. In Fig. 7, each set of bars represents a day during the testing period, and their height measures the total energy consumption for that day. The lighter color of each bar depicts the amount of energy consumed during off-peak hours, while the darker color represents energy consumption during peak hours. Even though on some days the total energy consumed by the DRL controller is higher than the baseline, the energy consumed during high price periods is significantly lower (50%). This behavior leads to a large reduction in overall cost (29%). The demand and cost values are summarized in Table 7. It is important to note that the baseline strategy is also shedding load using a rule-based algorithm, therefore the difference between DRL and the baseline can be read as extra-shed/differential shed.

Fig. 8 shows the detailed experimental results for the Shed experiment. The vertical gray bars in each panel represent the period with peak prices. Both controllers discharged the batteries during the peak price period, though the DRL controller discharged the battery more rapidly and by a larger amount. On average, at the end of the high price period, the DRL battery was left with less than 20% of its maximum charge, while the baseline battery was still at 50% (Fig. 8a). Further analysis revealed that the battery operation was largely responsible for the cost reduction of the DRL algorithm compared to the baseline.

Both controllers managed to reduce the chilled water consumption to zero for the duration of the peak prices, but the DRL engaged in more pre-cooling between 2pm – 4pm (12.4 kWh for the whole test) before the onset of the peak price period compared to the baseline (8 kWh) (Fig. 8b). The baseline controller also produced very sharp heating spikes which occurred all but one day around 7am, with different magnitude (Fig. 8c). In comparison, the DRL controller produced smaller morning heating spikes but caused some unexpected heating to happen during the interval corresponding to the peak price (Fig. 8c). This unexpected phenomenon is explored further in the Discussion in Section 6.2. Overall both controllers behaved poorly when it came to maintaining thermal comfort in the building during the peak price period (Fig. 8d), despite the relatively mild outdoor conditions (temp between 10.5 °C and 25.5 °C). The degree hour of temperature violations averaged per day was much higher for the baseline (≈ 4.92°C-h) than for the DRL controller (≈ 3.68°C-h). Further, when no heating was supplied by the DRL controller during the peak price period (on August 5, Fig. 8d), the indoor temperature was kept within the comfort band, suggesting that eliminating the unexpected heating during the Shed window would also improve comfort of the DRL cell. Active pre-cooling by DRL before the onset of the high price signal manifested as a sharp dip in the cell temperature right before the high price period.

An interesting behavior of the DRL controller emerged during the second day of the test (August 5). Since the day was foggy and PV generation was low, the DRL controller decided to charge the battery more than average, just before the Shed event. Hence, the additional availability of battery power allowed the DRL controller to reduce grid purchases during the high price window. In contrast the baseline did not foresee the issue and did not engage in any additional charging. Also, from Fig. 7, we can see that the DRL controller consumed less energy than the baseline cell on August 9. This can be attributed to the higher outdoor air temperatures and the strategy pursued by each controller. Both cells required a significant amount of cooling in the pre-event period, however the DRL compromised more on comfort and saved more energy compared to the baseline cell, relative to what happens during the other days. In these two days (08/05 and 08/09) the DRL controller is able to make better decisions than a rule based algorithm consistently reducing energy costs as well as power throughout the day.

5.3. Shift test

In the Shift experiment, the DRL controller was expected to increase the energy consumption and store thermal and electrical energy during the low (preferred) and medium price periods via pre-cooling and battery charging. Table 5 shows that, in the Shift experiment, the DRL controller reduced both cost and energy consumption compared to the baseline strategy, as anticipated. Table 8 summarizes the mean daily cost and average demand by the two controllers for different price periods. However, contrary to the expected behavior, the DRL controller consumed more energy during the peak period (i.e., it shifted less energy than the baseline strategy) but still reduced overall energy cost, because it significantly reduced the energy consumption during

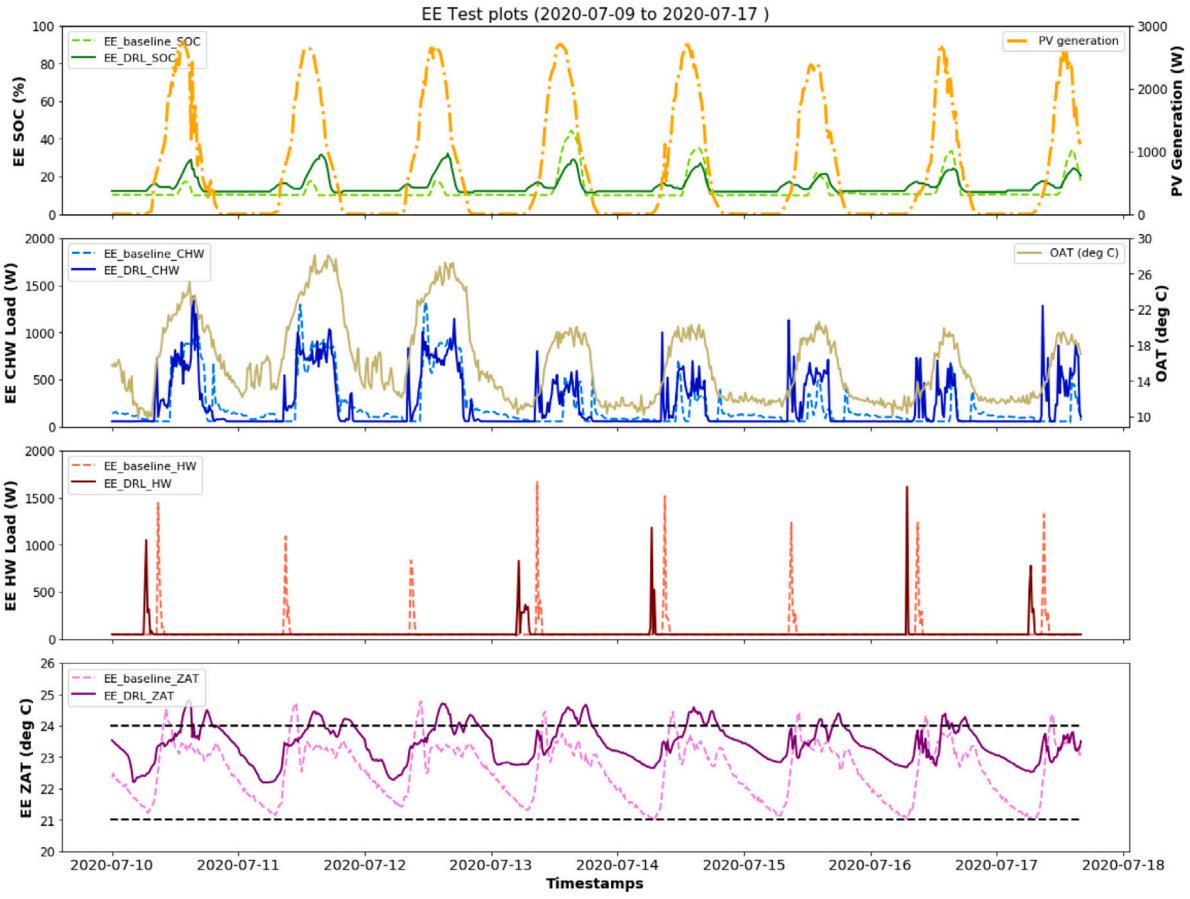


Fig. 6. EE test: (a) PV generation (gold), SOC of the battery in the baseline cell (light green) and SOC of the battery in the DRL cell (dark green); (b) chilled water power in the baseline cell (light blue) and the DRL cell (dark blue), dry bulb outdoor air temperature (yellow, right axis); (c) hot water power in the baseline cell (light red) and the DRL cell (dark red); (d) indoor dry bulb temperature in the baseline cell (light purple) and the DRL cell (dark purple).

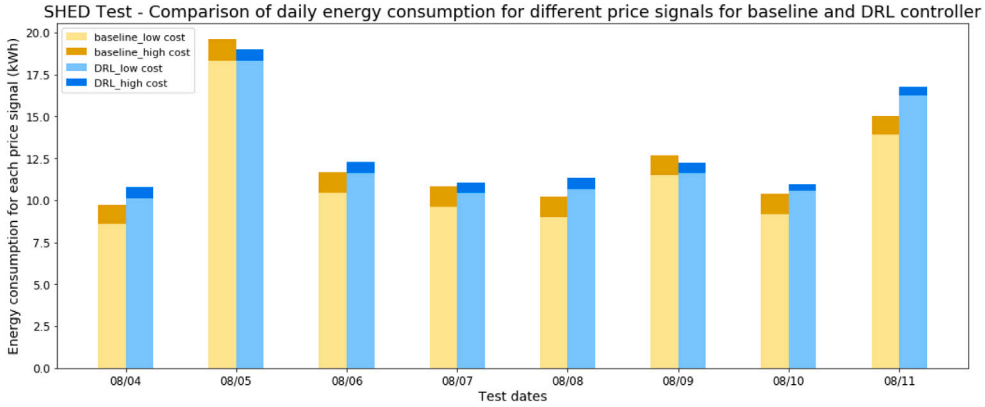


Fig. 7. Daily energy consumption for different prices for the baseline and DRL controller for the SHED control test.

the lower price periods. This result is interesting because the DRL algorithm achieved its objective of reducing cost, but it failed the implicit objective of shifting energy out of the peak period. These findings are further discussed in Section 6.3.

Fig. 9 shows details about the experiment. The vertical gray bars in each panel represent the mid peak (half gray bar) and high peak (full gray bar) periods. Both controllers discharged the batteries during the mid peak and peak periods (Fig. 9a), but the baseline controller discharged the battery by a larger amount. This is the first reason that explains the reduced shift in energy by the DRL controller, described above. The baseline controller used more chilled water and produced higher chilled water peaks than the DRL controller (Fig. 9b). On average, the DRL controller started cooling earlier, compared to the baseline, and continued cooling throughout the high peak period, causing an increase of demand during this period, but saving total energy and cost (Table 8). This is the second effect that limited the amount of energy shifted by the DRL algorithm. As it pertains to heating, the baseline controller created sharp spikes every early morning (Fig. 9c). In comparison, the DRL algorithm never used heating during the testing period. The DRL controller also provided better thermal comfort compared to the baseline (Fig. 9d), with 3.2 °C-h of temperature violations per day compared to 4.3 °C-h for the baseline. In addition, most of the comfort violations happened during the high price periods for the baseline cell, while they happened outside of that period for the DRL

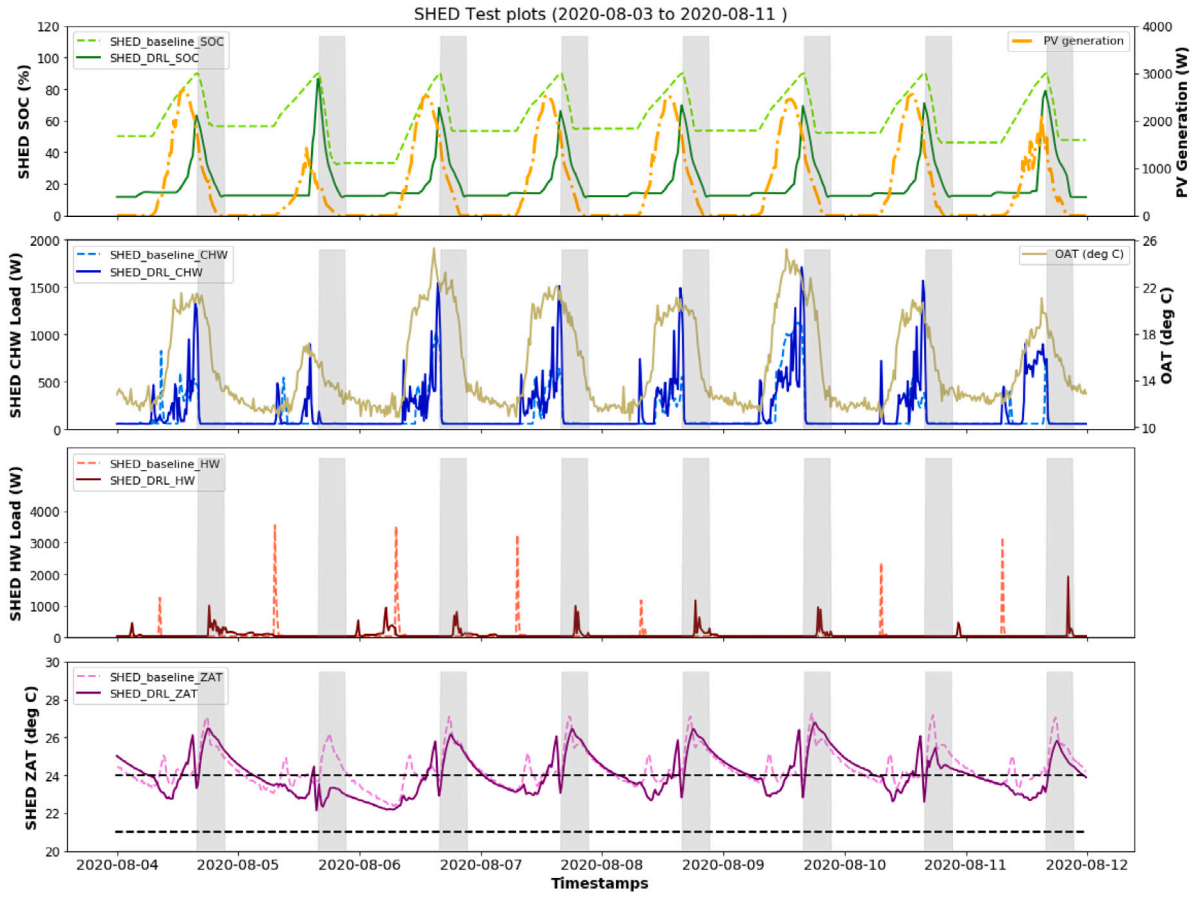


Fig. 8. Shed test: (a) PV generation (gold), SOC of the battery in the baseline cell (light green), and SOC of the battery in the DRL cell (dark green); (b) chilled water power in the baseline cell (light blue) and the DRL cell (dark blue), dry bulb outdoor air temperature (yellow, right axis); (c) hot water power in the baseline cell (light red) and the DRL cell (dark red); (d) indoor dry bulb temperature in the baseline cell (light purple) and the DRL cell (dark purple).

controller (Fig. 9d). Overlaying the comfort and chilled water profiles it is evident that DRL engaged in cooling during high price periods to maintain thermal comfort, in contrast to the baseline controller. This behavior suggests that DRL places a higher value on thermal comfort over cost savings during this period.

6. Discussion

6.1. Instability and variance of trained policies

Variation in trained policies across different random seeds is a common problem in DRL. Policies trained with different random seeds may generate a significant gap in performance and produce unstable results [66]. To investigate this variability, the performances of eight DRL agents, each trained with the same combination of hyper-parameters but with eight different seeds, were evaluated for each experiment. As described in Sections 4.2.3 and 4.2.4, the hyper-parameters were identified using the random search grid approach. For the Shift and Shed experiments, even out of these eight, only four converged to a realistic solution. While this in itself provides evidence of the significant impact of the random seeds, all the other controllers that had converged during training were evaluated further. The controllers were deployed in a simulated model of the building for (1) the duration of the actual FLEXLAB tests and (2) a whole year.

Fig. 10 shows the net energy consumption from the grid for the duration of the three experiments (EE, Shift, and Shed). While the green dots represent the actual energy consumed during the field experiment in FLEXLAB, the red dots are the energy consumption values estimated in the simulation environment using the same controller that was deployed in FLEXLAB (the best performer identified in the tuning process). Finally, the black dots represent the simulated energy consumption values using the policies that were generated using different random seeds during the training process. The actual energy consumption (in green) is significantly different from the energy consumption of the simulated model used for training the DRL controller (in red). Additionally, the random seeds seem to have a much higher impact in the EE experiment than for the Shift and Shed experiments. Our hypothesis for this effect is that it is likely due to the fixed price of electricity in EE test that make the algorithm converge more easily to some local optimal solution. In the EE test the difference between the best model (i.e., policy selected for the testbed) and the worst model is significant ($\approx$ 400 kWh). However, the small variability of the results for Shift and Shed may be misleading, as it is important to remember that only half of the experiments had converged. Fig. 11 shows the annual net energy purchased from the grid using the same controllers identified above, performing EE, Shed, and Shift experiments respectively. While the differences in values are not as prominent as in Fig. 10, it is clear that the seed has a significant impact on the actions taken by a DRL controller.

6.2. Mismatch between modeled and actual control behavior

When the trained DRL controller was deployed in the building, the actions generated by the controller differed from those taken during the simulation. Some of these occurred due to: using the wrong sensors, issues with mismatched units of measurement, or other factors, and they were able to be diagnosed and fixed immediately. However, there were more variations in the actions that produced significant

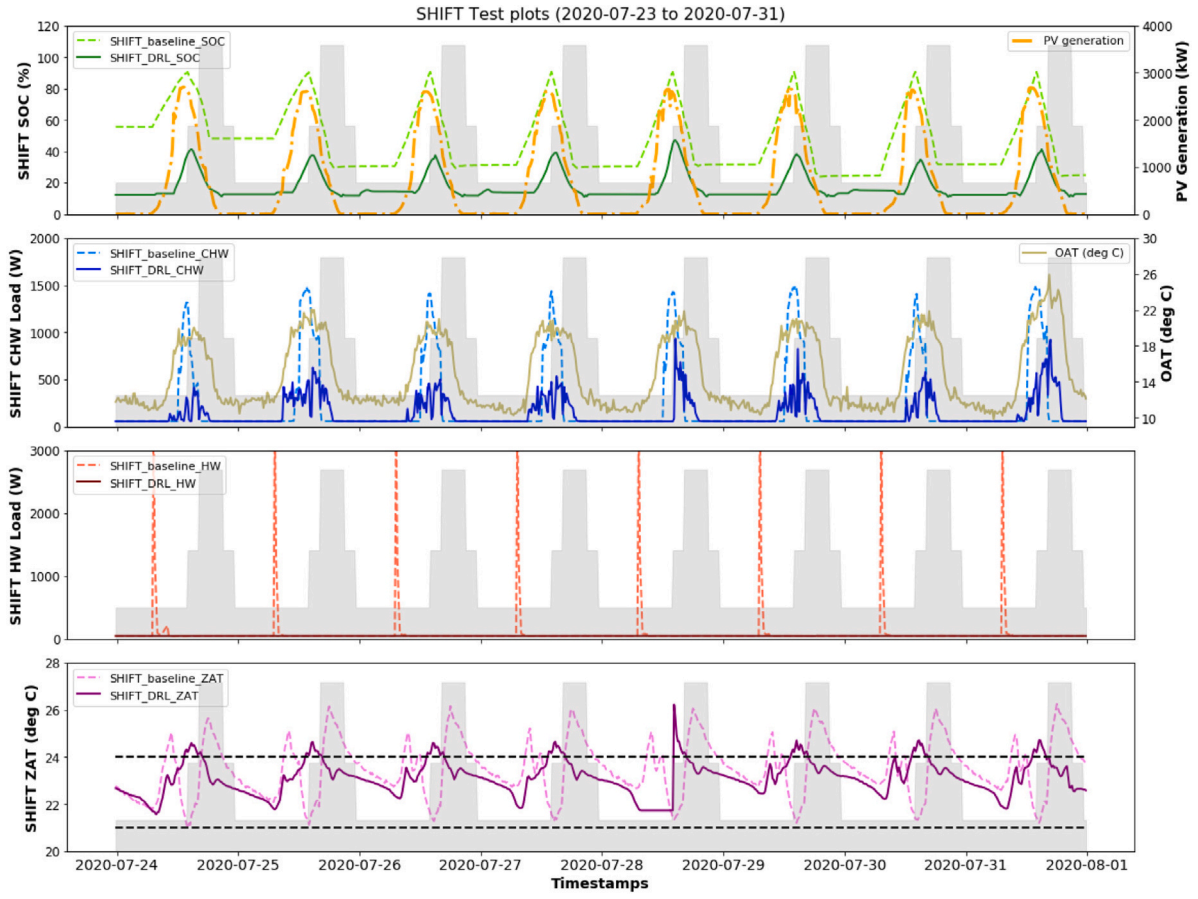


Fig. 9. Shift test: (a) PV generation (gold), SOC of the battery in the baseline cell (light green), and SOC of the battery in the DRL cell (dark green); (b) chilled water power in the baseline cell (light blue) and the DRL cell (dark blue), dry bulb outdoor air temperature (yellow, right axis); (c) hot water power in the baseline cell (light red) and the DRL cell (dark red); (d) indoor dry bulb temperature in the baseline cell (light purple) and the DRL cell (dark purple).

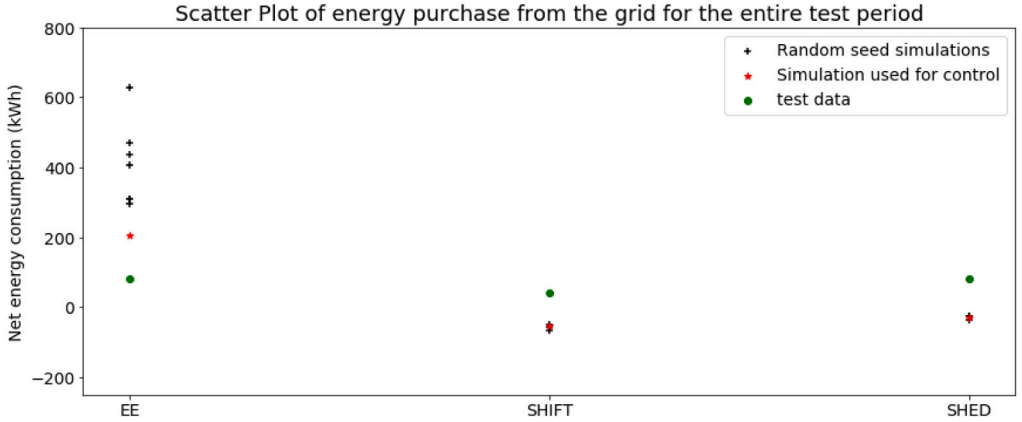


Fig. 10. Net energy consumption from the grid for simulated and test models.

differences in the relevant metrics (e.g., energy consumption, costs incurred), and upon further inspection it was determined that these happened due to inconsistencies between the simulated building energy model and the actual building. Such variations in actions were not present when the trained DRL controller was being trained and tested in the simulation environment and hence it was noticed only when it was deployed in the actual building. For example, the power consumption pattern of the supply air fan during cooling and the actual charge/discharge rate of the battery given a certain setpoint proved to be wrongly represented in the simulated building model. Once this issue was identified, the supply air fan was better characterized and the energy model was updated in the simulation (this, of course, meant retraining the DRL controller). However characterizing the stochastic battery charge/discharge behavior with respect to a given setpoint turned out to be quite challenging, and the DRL controller was used with this particular flaw built in. A more detailed evaluation of the impacts of these embedded inconsistencies has been presented in [67].

Similar inconsistencies that were overlooked during the model calibration also affected the results of our experiments. For example, in the EE test, we noticed that the DRL controller was unnecessarily heating the cell during early morning hours. Similarly during the Shed test, while the DRL controller performed well during the test, the actions

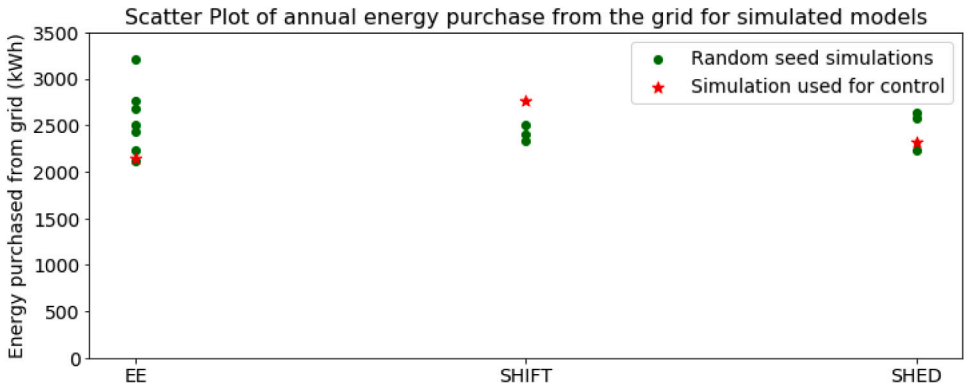


Fig. 11. Energy purchased from the grid for different simulated models for the entire year.

it generated produced diminished savings as the AHU started heating the zone during the high price period, reducing thermal comfort and increasing energy costs (Section 5.2). Upon further investigation (conducted after the testing), it was hypothesized that the modeling inconsistencies could be the most likely reason for this abnormal behavior. To gather evidence supporting this train of thought, the trained DRL agent was used to generate actions for the FLEXLAB simulated model used for training, subjected to the same external conditions (i.e., weather, occupancy, internal loads) of the actual Shed test. During the high price period, the agent generated similar supply air temperature and supply air flow rate setpoints as it did in the actual FLEXLAB test. However, the heating power required in the simulation, close to zero, was significantly lower than what was actually required in the real test. In other words there was an error in mapping the setpoints to the actual heating supplied by the equipment, probably due to simplifications in modeling the actual control sequences implemented in the underlying FLEXLAB infrastructure. The DRL agent expected the HVAC system to idle during the Shed period with the setpoints it generated, but that action, instead, caused significant heating to happen. This inconsistency in the heating energy model that the DRL agent was trained on negatively impacted the DRL controller's performance. The mismatch between HVAC and control operation in simulation models and real systems is common, therefore it is advisable to carefully map and calibrate the simulation with the actual behavior of the control system. In our case, we realized that the set of conditions we calibrated the building on did not adequately cover what was subsequently experienced during the tests. However, such calibration issues have major implications for practical scalability of DRL-based controllers.

6.3. Relative weight of cost and comfort and magnitude of price signal

During the Shift test, the DRL controller reduced the HVAC load significantly during the medium price period, which resulted in additional cooling (albeit for short periods) in the high price period, to reduce the thermal comfort penalty. This counterintuitive behavior was the result of the specific combination of the weights penalizing cost and comfort violations defined in the reward function (Section 4.2.2). The behavior was also indirectly affected by the prices selected for each period, since they determined cost. The DRL controller was successful in minimizing the total energy costs ($1.03 vs. $1.62 of the baseline controller) and obtained better comfort; however, it did not shift energy from the high price period to the other two periods, compared to the baseline Shed strategy. Since its behavior was driven by the magnitude of the prices, it is clear that the absolute value of the high price signal was not high enough to trigger the expected response, resulting in extra cooling load during the peak hours. As some utilities are moving toward dynamic pricing, this example highlights the challenges in designing price schemes that produce the desired grid response in a target building, particularly when the building employs algorithms that are unknown or unpredictable to the grid. In our example the DRL algorithm was successful in reducing cost for the building owner, but did not shift energy compared to the rule-based system.

Table 9
Comparison of metrics between shed test and shed test with higher penalty on thermal comfort.

Test	μ_v	σ_v	% of overshoot	ζ_v per day	Cost of daily energy
DRL (regular shed)	1.3	0.81	15	3.68	$ 2.46
DRL (more emphasis on thermal comfort)	1.25	0.58	6	1.88	$ 2.65

To understand the impact of the weight on thermal comfort, we conducted a test with a higher penalty being placed on thermal comfort violation, using the Shed test pricing. Results of this test are presented in Table 9. On this day, the DRL-controlled cell saw lower ZAT overshoot (1.88 °C-h) as compared to 3.68 °C-h overshoot during regular shed test days. However, this caused an increase in cost of energy purchased ($2.65) compared to cost of energy during regular shed days ($2.46). The experiment suggests that different trade-offs between comfort and cost can be achieved by tweaking the hyperparameters of the model.

6.4. Challenges and future work for deploying DRL-based controllers in buildings

The recent advances in DRL algorithms offer a unique opportunity to optimize the operation of complex building systems. However, there are still several challenges in developing, deploying, evaluating, and scaling DRL control algorithms in building systems.

First, training DRL controllers is a difficult task. Training DRL algorithm by directly interacting with real building systems may cause comfort issues and equipment damage and may take an unreasonable amount of time [68]. At the same time, developing a simulation model that adequately approximates the system is time consuming to develop, difficult to calibrate, and not scalable. And simpler models, such as gray-box models used in MPC [69], may be unable to fully capture the complex dynamic of real building systems. Thus, it is essential to develop DRL methods that require less training data and that use real historical data to ensure the scalability of this approach. Additional studies should investigate the minimum amount of data that needs to be collected to train such algorithms before deployment. Future research should also look into using transfer learning, imitation learning and also online learning techniques to address this challenge.

Second, as discussed in Section 6.1, the high variability of trained policies with different random seeds for the same combination set of hyper-parameters is a common problem for DRL algorithms. This issue is common while training neural networks and it is particularly relevant when dealing with incomplete observations of the states of a complex

system or reward functions that are hard to optimize. Future work should develop more stable DRL algorithms that improve the robustness to hyper-parameter tuning, as well as investigate the effectiveness of the random search grid as an approach for auto-tuning DRL parameters. In addition, including the random seed in the tuning process can enhance the selection of optimal policies.

Third, a proper definition of the reward functions is key to successfully finding a good control policy. Inadequate reward functions can result in unstable training and/or inappropriate policy. Therefore, more research is needed to analyze the impact of different definitions of reward functions (that optimize the same objective). This will help provide a more standardized definition of reward function that is well adapted for DRL frameworks.

Fourth, to promote adoption, the policies and behavior of the algorithm need to be easy to explain, particularly in the cases where the policy provide a nonstandard and unexpected solution to control the system. In addition, understanding the reasons behind control errors is important, to be able to create control algorithms that are robust to failures and thus reassure stakeholders, which is an essential step to increase the adoption of this technology.

Fifth, standardized building benchmarks are needed to properly evaluate the developed DRL methods. It is crucial to have standard environments where the DRL algorithms can be trained, to have well-defined metrics to quantify the performance of trained policies, and to have baseline DRL models against which the new algorithms can be evaluated. These benchmarks will help to speed incremental improvements and to evaluate the generalization of the proposed solutions to different buildings systems. By releasing and open-sourcing the environment and the DRL algorithm that was developed for this work, our aim is to provide the research community a benchmark for building systems that involves small commercial buildings with PV and battery storage. In the future, more of these benchmark will be needed to cover more applications (i.e., buildings systems and control objectives).

Lastly, future research should overcome the limitations of this study and extend its results by comparing different DRL algorithms and testing different control variables (e.g., controlling ZAT setpoints or chilled water valve position) in addition to the setpoints controlled in this experiment. Better comfort models, should be explored, to take advantage of more realistic assumptions about comfort adaptation. Additional field testing should be performed in a variety of building systems to make sure the results are generalizable.

7. Conclusion

This paper explores the use of a DRL approach to control a behind the meter DER system that consists of a proxy of a small office building with local PV solar generation and a battery unit. A simulation-based training framework was developed and used to train a DRL algorithm. This trained controller was then deployed in a testbed to evaluate its performance under three load flexibility modes (EE, Shift, and Shed). The results of this work, and prior published efforts show that DRL can be a promising solution for DER systems. As the research community transitions from simulation-based DRL research to field demonstrations, the results and the challenges described in this paper can be used to accelerate the efforts of future researchers and practitioners. The next steps in this research include improving the DRL training methods, evaluating different reward functions and testing alternative algorithms. These can help develop portable and scalable solutions that enable easier deployment of DRL in different buildings.

CRediT authorship contribution statement

Samir Touzani: Conceptualization, Methodology, Software, Validation, Investigation, Writing – original draft, Writing – review & editing, Supervision, Project administration. **Anand Krishnan Prakash:** Software, Validation, Formal analysis, Investigation, Data curation, Writing – original draft, Writing – review & editing, Visualization. **Zhe Wang:** Methodology, Software, Validation, Formal analysis, Investigation, Writing – original draft, Writing – review & editing. **Shreya Agarwal:** Formal analysis, Writing – original draft, Writing – review & editing, Visualization. **Marco Pritoni:** Methodology, Formal analysis, Writing – original draft, Writing – review & editing, Supervision. **Mariam Kiran:** Methodology, Writing – review & editing. **Richard Brown:** Writing – review & editing. **Jessica Granderson:** Conceptualization, Writing – review & editing, Supervision, Project administration, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported by the Assistant Secretary for Energy Efficiency and Renewable Energy, Building Technologies Office, of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231. The authors would like to thank Heather Goetsch, Cedar Blazek, the FLEXLAB team, Callie Clark and Christoph Gehbauer for their support. This research used the Lawrencium computational cluster resource provided by the IT Division at the Lawrence Berkeley National Laboratory (Supported by the Director, Office of Science, Office of Basic Energy Sciences, of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231).

References

[1] Bayram IS, Ustun TS. A survey on behind the meter energy management systems in smart grid. Renew Sustain Energy Rev 2017;72:1208–32.

[2] Foruzan E, Soh L-K, Asgarpoor S. Reinforcement learning approach for optimal distributed energy management in a microgrid. IEEE Trans Power Syst 2018;33(5):5749–58.

[3] Neukomm M, Nubbe V, Fares R. Grid-interactive efficient buildings. 2019, http://dx.doi.org/10.2172/1508212, URL https://www.osti.gov/biblio/1508212.

[4] Granderson J, Kramer H, Rui T, Brown R, Curtin C. Market brief: Customer-sited distributed energy resources. 2020, URL https://escholarship.org/uc/item/1xv3z9jb.

[5] Krishnan Prakash A, Zhang K, Gupta P, Blum D, Marshall M, Fierro G, Alstone P, Zoellick J, Brown R, Pritoni M. Solar+ optimizer: A model predictive control optimization platform for grid responsive building microgrids. Energies 2020;13(12). http://dx.doi.org/10.3390/en13123093, URL https://www.mdpi.com/1996-1073/13/12/3093.

[6] Bruno S, Giannoccaro G, La Scala M. A demand response implementation in tertiary buildings through model predictive control. IEEE Trans Ind Appl 2019;55(6):7052–61. http://dx.doi.org/10.1109/TIA.2019.2932963.

[7] Kim D, Braun J, Cai J, Fugate D. Development and experimental demonstration of a plug-and-play multiple RTU coordination control algorithm for small/medium commercial buildings. Energy Build 2015;107:279–93. http://dx.doi.org/10.1016/j.enbuild.2015.08.025, URL https://www.sciencedirect.com/science/article/pii/S0378778815302097.

[8] Bonthu RK, Pham H, Aguilera RP, Ha QP. Minimization of building energy cost by optimally managing PV and battery energy storage systems. In: 2017 20th international conference on electrical machines and systems (ICEMS). 2017, p. 1–6. http://dx.doi.org/10.1109/ICEMS.2017.8056442.

[9] Wang Z, Yang R, Wang L. Intelligent multi-agent control for integrated building and micro-grid systems. In: ISGT 2011. 2011, p. 1–7. http://dx.doi.org/10.1109/ISGT.2011.5759134.

[10] Reynolds J, Rezgui Y, Kwan A, Piriou S. A zone-level, building energy optimisation combining an artificial neural network, a genetic algorithm, and model predictive control. Energy 2018;151:729–39. http://dx.doi.org/10.1016/j.energy.2018.03.113, URL https://www.sciencedirect.com/science/article/pii/S036054421830522X.

[11] Macarulla M, Casals M, Forcada N, Gangolells M. Implementation of predictive control in a commercial building energy management system using neural networks. Energy Build 2017;151:511–9. http://dx.doi.org/10.1016/j.enbuild.2017.06.027, URL https://www.sciencedirect.com/science/article/pii/S0378778817300907.

[12] Drgoňa J, Arroyo J, Cupeiro Figueroa I, Blum D, Arendt K, Kim D, Ollé EP, Oravec J, Wetter M, Vrabie DL, Helsen L. All you need to know about model predictive control for buildings. Annu Rev Control 2020;50:190–232. http://dx.doi.org/10.1016/j.arcontrol.2020.09.001, URL https://www.sciencedirect.com/science/article/pii/S1367578820300584.

[13] Vázquez-Canteli JR, Nagy Z. Reinforcement learning for demand response: A review of algorithms and modeling techniques. Appl Energy 2019;235:1072–89. http://dx.doi.org/10.1016/j.apenergy.2018.11.002, URL http://www.sciencedirect.com/science/article/pii/S0306261918317082.

[14] Sutton RS, Barto AG. Reinforcement learning: An introduction. USA: A Bradford Book; 2018.

[15] Wang Z, Hong T. Reinforcement learning for building controls: The opportunities and challenges. Appl Energy 2020;269:115036. http://dx.doi.org/10.1016/j.apenergy.2020.115036, URL http://www.sciencedirect.com/science/article/pii/S0306261920305481.

[16] White CC, White DJ. Markov decision processes. European J Oper Res 1989;39:1–16, URL https://doi.org/10.1016/0377-2217(89)90348-2.

[17] Silver D, Schrittwieser J, Simonyan K, Antonoglou I, Huang A, Guez A, Hubert T, Baker L, Lai M, Bolton A, et al. Mastering the game of go without human knowledge. Nature 2017;550(7676):354–9.

[18] Mocanu E, Mocanu DC, Nguyen PH, Liotta A, Webber ME, Gibescu M, Slootweg JG. On-line building energy optimization using deep reinforcement learning. IEEE Trans Smart Grid 2019;10(4):3698–708.

[19] Wei T, Wang Y, Zhu Q. Deep reinforcement learning for building HVAC control. In: Proceedings of the 54th annual design automation conference 2017. 2017; p. 1–6.

[20] Mnih V, Kavukcuoglu K, Silver D, Rusu AA, Veness J, Bellemare MG, Graves A, Riedmiller M, Fidjeland AK, Ostrovski G, Petersen S, Beattie C, Sadik A, Antonoglou I, King H, Kumaran D, Wierstra D, Legg S, Hassabis D. Human-level control through deep reinforcement learning. Nature 2015;518(7540):529–33. http://dx.doi.org/10.1038/nature14236.

[21] Lillicrap TP, Hunt JJ, Pritzel A, Heess N, Erez T, Tassa Y, Silver D, Wierstra D. Continuous control with deep reinforcement learning. 2015, arXiv preprint arXiv:1509.02971.

[22] Silver D, Lever G, Heess N, Degris T, Wierstra D, Riedmiller M. Deterministic policy gradient algorithms. In: Proceedings of the 31st international conference on international conference on machine learning - Volume 32. ICML'14, JMLR.org; 2014, p. I–387–I–395.

[23] Henderson P, Islam R, Bachman P, Pineau J, Precup D, Meger D. Deep reinforcement learning that matters. In: Proceedings of the AAAI conference on artificial intelligence, Vol. 32. 2018, (1). URL https://ojs.aaai.org/index.php/AAAI/article/view/11694.

[24] Liang E, Liaw R, Nishihara R, Moritz P, Fox R, Goldberg K, Gonzalez J, Jordan M, Stoica I. RLlib: Abstractions for distributed reinforcement learning. In: Dy J, Krause A, editors. Proceedings of the 35th international conference on machine learning. Proceedings of machine learning research, vol. 80, Stockholmsmässan, Stockholm Sweden: PMLR; 2018, p. 3053–62, URL http://proceedings.mlr.press/v80/liang18b.html.

[25] Mason K, Grijalva S. A review of reinforcement learning for autonomous building energy management. Comput Electr Eng 2019;78:300–12. http://dx.doi.org/10.1016/j.compeleceng.2019.07.019, URL http://www.sciencedirect.com/science/article/pii/S0045790618333421.

[26] Yang T, Zhao L, Li W, Zomaya AY. Reinforcement learning in sustainable energy and electric systems: a survey. Annu Rev Control 2020;49:145–63. http://dx.doi.org/10.1016/j.arcontrol.2020.03.001, URL https://www.sciencedirect.com/science/article/pii/S1367578820300079.

[27] Wang Y, Velswamy K, Huang B. A long-short term memory recurrent neural network based reinforcement learning controller for office heating ventilation and air conditioning systems. Processes 2017;5(3). http://dx.doi.org/10.3390/pr5030046, URL https://www.mdpi.com/2227-9717/5/3/46.

[28] Jia R, Jin M, Sun K, Hong T, Spanos C. Advanced building control via deep reinforcement learning. Energy Procedia 2019;158:6158–63. http://dx.doi.org/10.1016/j.egypro.2019.01.494, Innovative Solutions for Energy Transitions. URL http://www.sciencedirect.com/science/article/pii/S187661021930517X.

[29] Wei T, Wang Y, Zhu Q. Deep reinforcement learning for building HVAC control. In: 2017 54th ACM/EDAC/IEEE design automation conference (DAC). 2017, p. 1–6.

[30] Brandi S, Piscitelli MS, Martellacci M, Capozzoli A. Deep reinforcement learning to optimise indoor temperature control and heating energy consumption in buildings. Energy Build 2020;224:110225. http://dx.doi.org/10.1016/j.enbuild.2020.110225, URL http://www.sciencedirect.com/science/article/pii/S0378778820308963.

[31] Yoon YR, Moon HJ. Performance based thermal comfort control (PTCC) using deep reinforcement learning for space cooling. Energy Build 2019;203:109420. http://dx.doi.org/10.1016/j.enbuild.2019.109420, URL http://www.sciencedirect.com/science/article/pii/S0378778819310692.

[32] Gao G, Li J, Wen Y. Energy-efficient thermal comfort control in smart buildings via deep reinforcement learning. 2019, URL arXiv:1901.04693.

[33] Azuatalam D, Lee W-L, de Nijs F, Liebman A. Reinforcement learning for whole-building HVAC control and demand response. Energy AI 2020;2:100020. http://dx.doi.org/10.1016/j.egyai.2020.100020, URL http://www.sciencedirect.com/science/article/pii/S2666546820300203.

[34] Nagy A, Kazmi H, Cheaib F, Driesen J. Deep reinforcement learning for optimal control of space heating. In: 4th building simulation and optimization conference. 2018, URL http://www.ibpsa.org/proceedings/BSO2018/1C-4.pdf.

[35] Lee S, Choi D-H. Energy management of smart home with home appliances, energy storage system and electric vehicle: A hierarchical deep reinforcement learning approach. Sensors 2020;20(7). http://dx.doi.org/10.3390/s20072157, URL https://www.mdpi.com/1424-8220/20/7/2157.

[36] Chen B, Cai Z, Bergés M. Gnu-rl: A precocial reinforcement learning solution for building hvac control using a differentiable mpc policy. In: Proceedings of the 6th ACM international conference on systems for energy-efficient buildings, cities, and transportation. 2019; p. 316–25.

[37] Zhang Z, Lam KP. Practical implementation and evaluation of deep reinforcement learning control for a radiant heating system. In: Proceedings of the 5th conference on systems for built environments. BuildSys '18, New York, NY, USA: Association for Computing Machinery; 2018, p. 148–57. http://dx.doi.org/10.1145/3276774.3276775.

[38] Moriyama T, De Magistris G, Tatsubori M, Pham T-H, Munawar A, Tachibana R. Reinforcement learning testbed for power-consumption optimization. In: Li L, Hasegawa K, Tanaka S, editors. Methods and applications for modeling and simulation of complex systems. Singapore: Springer Singapore; 2018, p. 45–59.

[39] Zhang Z, Chong A, Pan Y, Zhang C, Lu S, Lam KP. A deep reinforcement learning approach to using whole building energy model for HVAC optimal control. In: ASHRAE and IBPSA-USA simbuild building performance modeling conference. 2018, URL https://www.researchgate.net/publication/326711617_A_Deep_Reinforcement_Learning_Approach_to_Using_Whole_Building_Energy_Model_For_HVAC_Optimal_Control.

[40] Zhang Z, Zhang C, Lam K. A deep reinforcement learning method for model-based optimal control of HVAC systems. 2018, p. 397–402. http://dx.doi.org/10.14305/ibpc.2018.ec-1.01.

[41] Zhang Z, Chong A, Pan Y, Zhang C, Lam KP. Whole building energy model for HVAC optimal control: A practical framework based on deep reinforcement learning. Energy Build 2019;199:472–90. http://dx.doi.org/10.1016/j.enbuild.2019.07.029, URL http://www.sciencedirect.com/science/article/pii/S0378778818330858.

[42] Li Y, Wen Y, Tao D, Guan K. Transforming cooling optimization for green data center via deep reinforcement learning. IEEE Trans Cybern 2020;50(5):2002–13.

[43] Kazmi H, Mehmood F, Lodeweyckx S, Driesen J. Gigawatt-hour scale savings on a budget of zero: Deep reinforcement learning based optimal control of hot water systems. Energy 2018;144:159–68. http://dx.doi.org/10.1016/j.energy.2017.12.019, URL http://www.sciencedirect.com/science/article/pii/S0360544217320388.

[44] Wu P, Partridge J, Bucknall R. Cost-effective reinforcement learning energy management for plug-in hybrid fuel cell and battery ships. Appl Energy 2020;275:115258. http://dx.doi.org/10.1016/j.apenergy.2020.115258, URL http://www.sciencedirect.com/science/article/pii/S0306261920307704.

[45] Qiu S, Li Z, Li Z, Li J, Long S, Li X. Model-free control method based on reinforcement learning for building cooling water systems: Validation by measured data-based simulation. Energy Build 2020;218:110055. http://dx.doi.org/10.1016/j.enbuild.2020.110055, URL http://www.sciencedirect.com/science/article/pii/S0378778819339945.

[46] Soares A, Geysen D, Spiessens F, Ectors D, De Somer O, Vanthournout K. Using reinforcement learning for maximizing residential self-consumption – Results from a field test. Energy Build 2020;207:109608. http://dx.doi.org/10.1016/j.enbuild.2019.109608, URL http://www.sciencedirect.com/science/article/pii/S037877881930934X.

[47] Yu L, Xie W, Xie D, Zou Y, Zhang D, Sun Z, Zhang L, Zhang Y, Jiang T. Deep reinforcement learning for smart home energy management. IEEE Internet Things J 2020;7(4):2751–62. http://dx.doi.org/10.1109/JIOT.2019.2957289.

[48] Alfaverh F, Denaï M, Sun Y. Demand response strategy based on reinforcement learning and fuzzy reasoning for home energy management. IEEE Access 2020;8:39310–21. http://dx.doi.org/10.1109/ACCESS.2020.2974286.

[49] Schreiber T, Eschweiler S, Baranski M, Müller D. Application of two promising Reinforcement Learning algorithms for load shifting in a cooling supply system. Energy Build 2020;229:110490. http://dx.doi.org/10.1016/j.enbuild.2020.110490, URL http://www.sciencedirect.com/science/article/pii/S0378778820320922.

[50] Hua H, Qin Y, Hao C, Cao J. Optimal energy management strategies for energy Internet via deep reinforcement learning approach. Appl Energy 2019;239:598–609.

[51] BL Energy Technologies Area. FLEXLAB: Advanced integrated building & grid technologies testbed. 2020, URL https://flexlab.lbl.gov/.

[52] Yu L, Qin S, Zhang M, Shen C, Jiang T, Guan X. Deep reinforcement learning for smart building energy management: A survey. 2020, arXiv:2008.05074.

[53] EI Administration. 2012 CBECS survey data - microdata. 2020, https://www.eia.gov/consumption/commercial/data/2012/index.php?view=microdata. [Online; accessed 17-December-2020].

[54] P Gas, E Company. Electric schedule B-19. 2020, https://www.pge.com/tariffs/assets/pdf/tariffbook/ELEC_SCHEDS_B-19.pdf. [Online; accessed 16-December-2020].

[55] Vasudevan J, Swarup KS. Price based demand response strategy considering load priorities. In: 2016 IEEE 6th international conference on power systems (ICPS). 2016, p. 1–6. http://dx.doi.org/10.1109/ICPES.2016.7584019.

[56] Asadinejad A, Tomsovic K. Optimal use of incentive and price based demand response to reduce costs and price volatility. Electr Power Syst Res 2017;144:215–23. http://dx.doi.org/10.1016/j.epsr.2016.12.012, URL https://www.sciencedirect.com/science/article/pii/S0378779616305259.

[57] Liu J, Yin R, Pritoni M, Piette MA, Neukomm M. Developing and evaluating metrics for demand flexibility in buildings: Comparing simulations and field data. 2020, URL https://escholarship.org/uc/item/2d93p636.

[58] ASHRAE. New guideline on standardized advanced sequences of operation for common HVAC systems. 2018, URL https://www.ashrae.org/news/esociety/new-guideline-on-standardized-advanced-/sequences-of-operation-for-common-hvac-systems.

[59] Nouidui T, Wetter M, Zuo W. Functional mock-up unit for co-simulation import in EnergyPlus. J Build Perform Simul 2014;7(3):192–202.

[60] Gehbauer C, Mueller J, Swenson T, Vrettos E. Photovoltaic and behind-the-meter battery storage: Advanced smart inverter controls and field demonstration. 2020.

[61] Andersson C, Åkesson J, Führer C. Pyfmi: A Python package for simulation of coupled dynamic models with the functional mock-up interface. Centre for Mathematical Sciences, Lund University Lund; 2016.

[62] Zhang Z, Chong A, Pan Y, Zhang C, Lam KP. Whole building energy model for HVAC optimal control: A practical framework based on deep reinforcement learning. Energy Build 2019;199:472–90.

[63] Morris MD. Factorial sampling plans for preliminary computational experiments. Technometrics 1991;33(2):161–74.

[64] Bergstra J, Bengio Y. Random search for hyper-parameter optimization. J Mach Learn Res 2012;13(2).

[65] Kingma DP, Ba J. Adam: A method for stochastic optimization. 2014, arXiv preprint arXiv:1412.6980.

[66] Islam R, Henderson P, Gomrokchi M, Precup D. Reproducibility of benchmarked deep reinforcement learning tasks for continuous control. 2017, arXiv preprint arXiv:1708.04133.

[67] Prakash AK, Touzani S, Kiran M, Agarwal S, Pritoni M, Granderson J. Deep reinforcement learning in buildings: Implicit assumptions and their impact. In: Proceedings of the 1st international workshop on reinforcement learning for energy management in buildings & cities. RLEM'20, New York, NY, USA: Association for Computing Machinery; 2020, p. 48–51. http://dx.doi.org/10.1145/3427773.3427868.

[68] Tan Y, Yang J, Chen X, Song Q, Chen Y, Ye Z, Su Z. Sim-to-real optimization of complex real world mobile network with imperfect information via deep reinforcement learning from self-play. 2018, arXiv:1802.06416.

[69] Sohlberg B, Jacobsen E. Grey box modelling - branches and experiences. IFAC Proc Vol 2008;41(2):11415–20. http://dx.doi.org/10.3182/20080706-5-KR-1001.01934, 17th IFAC World Congress. URL http://www.sciencedirect.com/science/article/pii/S1474667016408025.