Two-Stage Reinforcement Learning Policy Search for Grid-Interactive Building Control

Xiangyu Zhang, Yue Chen, Andrey Bernstein, Rohit Chintala, Peter Graf, Xin Jin and David Biagioni

***Abstract*—This paper develops an intelligent grid-interactive building controller, which optimizes building operation during both normal hours and demand response (DR) events. To avoid costly on-demand computation and to adapt to non-linear building models, the controller utilizes reinforcement learning (RL) and makes real-time decisions based on a near-optimal control policy. Learning such a policy typically amounts to solving a hard non-convex optimization problem. We propose to address this problem with a novel *global-local policy search* method. In the first stage, an RL algorithm based on zero-order gradient estimation is leveraged to search for the optimal policy globally, due to its scalability and the potential to escape some poor performing local optima. The obtained policy is then fine-tuned locally to bring the first-stage solution closer to that of the original unsmoothed problem. Experiments on a simulated five-zone commercial building demonstrate the advantages of the proposed method over existing learning approaches. They also show that the learned control policy outperforms a pragmatic linear model predictive controller (MPC) and approaches the performance of an oracle MPC in testing scenarios. Using a state-of-the-art advanced computing system, we demonstrate that the controller can be learned and deployed within hours of training.**

***Index Terms*—Reinforcement learning, zero-order gradient estimation, demand response, smart building.**

I. Introduction

As the buildings sector represents over 70% of the total U.S. electricity consumption [1], it offers a great amount of untapped demand-side resources to tackle many critical grid-side problems. By and large, buildings can provide grid services to both the transmission and distribution operational levels [2] in the form of demand modification [3] and ancillary services [4], [5]. Specifically, among commercial buildings, about 33% of the electricity usage is for the heating, ventilation, and air-conditioning (HVAC) systems [6]. The high energy consumption, together with the potential flexibility provided by building thermal mass, makes the HVAC system a good candidate for grid-interactive control. However, due to the complexity of the building thermal dynamics and the strong inter-temporal dependency of the control problem, proper design of an intelligent building HVAC controller that is effective, practical, and economic for massive deployment, is challenging.

Currently, mainstream grid-interactive building HVAC controllers in practice and under research are based on either direct load control (DLC) or model predictive control (MPC). DLC enables utilities or aggregators to directly and remotely control customers' HVAC systems as needed and is simple and cheap for massive implementation, see [7] for a real-world example. Nevertheless, DLC does not directly consider specific building thermal condition and thus might jeopardize occupant comfort. MPC, on the other hand, can be used for building optimal control, and it is widely studied in building applications. To improve computational efficiency during real-time control, linear building models such as [8] are used in many cases. To handle the stochastic environmental disturbances, occupancy forecasts are integrated into MPC [9], and [10] proposes a tube-based MPC that achieves good performance as a robust controller but requires less computational resources. For grid-interactive control, MPC has been utilized for participation in electricity market [11] and peak-shaving demand response (DR) programs [12], [13]. These building-MPC studies are by no means exhaustive; for a more in-depth and systematic review, we refer interested readers to [14], [15]. Though MPC can achieve multi-objective (building + grid) optimal control and can theoretically guarantee the optimality in many cases, its massive deployment can be challenging. As summarized in [16], MPC requires 1) a carefully designed building-specific simplified (i.e., linearized) model, 2) forecast modules for environmental disturbances, and 3) on-demand computation to solve optimization problems during real time control; all these features lead to high implementation costs (i.e., hardware, software, and modeling) and make this controller prohibitive for building owners.

Reinforcement learning (RL), on the other hand, enables "offline" controller training using more accurate nonlinear building models before it is deployed and can avoid expensive on-demand computation. RL is a successful alternative for many optimal sequential decision-making problems [17]–[19], and has been investigated for many building energy management applications, including load-level control [20]–[22], residential building control [23], [24] and aggregated buildings control [25]. For commercial buildings control, RL algorithms, including the deep Q-network (DQN) and asynchronous advantage actor-critic (A3C), are utilized for energy-saving while maintaining indoor comfort [26], [27]. However, the discrete action spaces they employed usually require

This work was authored by the National Renewable Energy Laboratory, operated by Alliance for Sustainable Energy, LLC, for the U.S. Department of Energy (DOE) under Contract No. DE-AC36-08GO28308. Funding provided by the U.S. Department of Energy Office of Energy Efficiency and Renewable Energy Building Technologies Office. The views expressed in the article do not necessarily represent the views of the DOE or the U.S. Government. The U.S. Government retains and the publisher, by accepting the article for publication, acknowledges that the U.S. Government retains a nonexclusive, paid-up, irrevocable, worldwide license to publish or reproduce the published form of this work, or allow others to do so, for U.S. Government purposes.

This research was performed using computational resources sponsored by the Department of Energy's Office of Energy Efficiency and Renewable Energy and located at the National Renewable Energy Laboratory.

careful discretization to achieve good control performance and are more susceptible to the "curse of dimensionality" when applied to multi-zone control [26, Section 3.3]. Though [26] proposed a heuristic way to battle this issue, their approach of training separate neural network for each thermal zone does not allow multi-zone coordination, which is a valuable behavior under DR events when the cooling capability is limited. To use continuous action space, which greatly increases the policy search space and problem complexity, a Zap Q-learning method is leveraged in [28], though its application does not consider multi-zone building control. In addition, most prior building-RL studies are building-centric and do not consider enabling a building's grid-interactive feature. The few exceptions include [16], which studies a multiple air-conditioning (AC) units coordination problem under DR events, and [29], which shares the most similar application with our present paper; the performance of the algorithm used therein will be compared to that of the present paper in the following. Moreover, many studies develop controllers which, similar to MPC, require forecasts of some hard-to-obtain exogenous data (e.g., solar radiation in each building thermal zone [26], [27], [29]). This is undesirable in real-world applications due to the requirement of a sophisticated sensor network and dedicated forecasting modules for different environment variables. Finally, the literature is mostly focused on application of a given/off-the-shelf RL algorithm to the building control problem, without discussing algorithmic questions regarding how to achieve a faster convergence to a good policy in this application domain.

Policy search in RL typically amounts to solving hard non-convex optimization problems, even for a linear quadratic regulator (LQR) control problem, see Appendix.B of [30]. Though [30] provides a provable guarantee of global convergence for searching an RL policy for LQR, the required *gradient domination* condition is hard to generalize to other problems. Therefore, when solving complex control problems with high dimensional search spaces, complex dynamics/constraints, and multiple objectives, if not properly configured, RL can oftentimes converge to a trained policy that is a poor-performing local optimum. Additionally, although momentum-based optimization algorithms (e.g., Nesterov accelerated gradient [31] and Adam [32]) have the potential to escape local optima, their effective application to RL policy search requires careful tuning of the learning step size and momentum suitable for a specific problem.

In order to achieve faster convergence to a better control policy, this paper extends the preliminary idea first presented in [33] and proposes a novel *two-stage global-local RL policy search method* that combines the complementary advantages of a zero-order gradient estimation (ZOE) approach [34], [35] and a policy gradient algorithm [17, Chapter 13], [36]. Prior to our work, there was an attempt [37] to combine such two types of RL algorithms. However, the mixed policy update approach used in that work causes potential interference between the two and thus did not deliver satisfactory results. In contrast, our proposed staged combination allows both algorithms to work independently and leverage their strengths to the greatest extent at appropriate time.

Specifically, this paper provides the following contributions:

1) From the RL algorithmic aspect, we first illustrate the convergence features of ZOE-based methods and how they can help avoid convergence to some poor-performing policies. Then, an intrinsic caveat of such algorithms is identified, i.e., the converged policy is not optimal to the original problem due to the function smoothing. To address this, the two-stage learning approach is proposed, in which a policy gradient method is used to fine-tune the policy learned by the ZOE-based method and bring it closer to the true optimum, i.e., a better control policy.

2) From the application aspect, using the most complex single building control problem studied in RL-building literature, we demonstrate the benefits of the proposed two-stage learning approach in a simulated environment. The performance of the RL controller learned via the proposed method is compared with those trained by standard RL approaches and two MPC baseline controllers, and the experiment results demonstrate the efficacy and necessity of the proposed method. In addition, some aspects regarding real-world implementation such as RL controller's input design and training RL controllers on an advanced computing platform are also discussed.

The remainder of the paper is organized as follows. Section II defines the mathematical formulation of the optimal building control problem. Section III describes the design of the RL controller and Section IV discusses the proposed two-stage policy search method for training the controller. Case study on a five-zone commercial building is provided in Section V and finally Section VI concludes the paper and some discussions are provided.

II. Problem Formulation

A. Multi-Zone Commercial Building HVAC Control

Commercial buildings usually have multiple thermal zones $i \in \mathcal{N} = \{1, 2, ..., N\}$, and the thermal dynamics for each zone is relatively independent from others. For buildings with a centralized AC system, typically, the chiller discharge air temperature (T^{da}) and zone-specific cooling air flow rates ($\dot{m}^i, i \in \mathcal{N}$) are controlled to provide a comfortable indoor environment. We define $\mathbf{a}_t = [\dot{m}_t^1, \dot{m}_t^2, ..., \dot{m}_t^N, T_t^{da}]^\top \in \mathcal{A} \subset \mathbb{R}^{N+1}$ to represent values of all control variables at step $t \in \mathcal{T}$, where $\mathcal{A}$ describes a constrained action space due to operation limits (i.e., $T_t^{da} \in [\underline{T}^{da}, \overline{T}^{da}]$ and $\dot{m}_t^i \in [\underline{\dot{m}_t^i}, \overline{\dot{m}_t^i}]$ for $i \in \mathcal{N}$) and the set $\mathcal{T}$ collects all control intervals in the control horizon. Through these control variables, this paper aims to develop an intelligent controller to a) optimize AC system's energy consumption while b) minimizing the indoor discomfort and c) providing grid services. Though cooling scenarios are focused in this paper, the proposed method can also be applied in the building heating season.

In this paper, to provide grid service, the building will participate in an incentive-based DR program [38], by reducing the HVAC power consumption below a certain limit for a few hours upon receiving the grid signal.

B. Optimization Formulation

The above-mentioned optimal sequential control problem is formulated as a multi-objective optimization given by:

$$\begin{aligned} \underset{\mathbf{a}_t \in \mathcal{A}, \forall t}{\text{minimize}} \quad & \sum_{t \in \mathcal{T}} \mathbf{w}_t^\top \mathbf{C}_t \\ \text{subject to} \quad & \mathbf{T}_{t+1} = \mathcal{F}\left(\mathbf{T}_t, \mathbf{a}_t, \boldsymbol{\varrho}_t\right) \quad (\forall t \in \mathcal{T}), \end{aligned} \tag{1}$$

in which $\mathbf{C}_t := [\kappa_1 \sum_{i \in \mathcal{N}} \mathcal{D}(T_t^i), \kappa_2 \mathcal{E}_t, \kappa_3 \mathcal{V}_t]^\top$ is the single step cost vector, κ_i with $i \in \{1, 2, 3\}$ are nominal monetizing factors for the three objectives, and $\mathbf{w}_t := [w_t^1, w_t^2, w_t^3]^\top$ is the weight vector with $\mathbf{w}_t^\top \mathbf{1} = 1$. Specifically, the thermal discomfort cost $\mathcal{D}(T_t^i)$ measures the temperature deviation from a pre-defined comfort band $[\underline{T}^i, \overline{T}^i]$ with a piece-wise function:

$$\mathcal{D}(T_t^i) := \begin{cases} \max(T_t^i - \overline{T}^i, (T_t^i - \overline{T}^i)^2) & (T_t^i > \overline{T}^i) \\ \max(\underline{T}^i - T_t^i, (\underline{T}^i - T_t^i)^2) & (T_t^i < \underline{T}^i) \\ 0.0 & (\text{else}) \end{cases} \tag{2}$$

where T_t^i is the indoor temperature of zone i at step t.

Second, the energy consumed at step t is calculated by:

$$\mathcal{E}_t := P(\mathbf{a}_t, T_t^{out}) \Delta t, \tag{3}$$

in which Δt is the length of control interval and the HVAC system power consumption $P(\mathbf{a}_t, T_t^{out})$ is given by:

$$P(\mathbf{a}_t, T_t^{out}) := a(T_t^{out} - T_t^{da}) \sum_{i=1}^{N} \dot{m}_t^i + b(\sum_{i=1}^{N} \dot{m}_t^i)^3 + c. \tag{4}$$

The first term in (4) describes the chiller power [39] and the rest depicts the fan power; T_t^{out} is the outdoor temperature and a, b and c are known constants.

Third, for grid-interactive control, a penalty on the power limit violation is given by:

$$\mathcal{V}_t := \begin{cases} (P(\mathbf{a}_t, T_t^{out}) - \overline{P}_t)^2 & (P(\mathbf{a}_t, T_t^{out}) \geq \overline{P}_t) \\ 0 & (\text{else}) \end{cases} \tag{5}$$

which is zero if the HVAC system's power is strictly below a given limit $\overline{P}_t$; otherwise, the cost increases quadratically with the exceeding amount. Typically, during normal hours, $\overline{P}_t$ is set to a higher value aiming at reducing the monthly demand charge [40]; and during a DR event ($t \in \mathcal{T}_{DR}$), $\overline{P}_t$ is reduced drastically as per the building-utility DR contract in order to achieve the load reduction objective.

Besides the feasibility of control variables (i.e. $\mathbf{a}_t \in \mathcal{A}$), equality constraints are included to enforce the building thermal dynamics. In particular, $\mathcal{F}$ represents the building dynamics model that can be obtained in a variety of ways. In this paper, we leverage a data-driven approach using an Auto-Regressive model with eXogenous variables (ARX), yielding a generally non-linear mapping $\mathcal{F}$. The process of obtaining such a model is not the focus of this paper and is thus omitted; we refer the interested reader to [41] for details. $\mathbf{T}_t \in \mathbb{R}^N$ collects the indoor temperature of all zones and $\boldsymbol{\varrho}_t$ represents the environmental disturbances (e.g., outdoor temperature and influences from solar radiation and building occupancy).

Typically, MPC is used to solve problem (1) in a receding horizon manner: at every step, using updated system states and the forecasts of exogenous inputs, MPC solves (1), implements the first step of the optimized values, discards the rest, and moves on to the next step to repeat the process. In the next sections, we propose an alternative RL-based approach to solve (1), investigating if the RL approach can achieve the MPC control performance and address some of the MPC's weaknesses mentioned in Section I.

III. Reinforcement Learning Controller

In general, in an RL formulation, an agent learns an action strategy, called the *policy*, to guide it for a sequential interaction with an environment in order to maximize the expected rewards it receives over a finite or infinite horizon. The goal of RL training is to search for the best policy over a parameter space that defines all possible policies. Necessary preliminary knowledge on RL and its mathematical formalism is provided next; interested readers can also refer to [17, Chapter 3] for a more in-depth understanding.

A. Markov Decision Process and RL

Mathematically, to formalize RL, *Markov Decision Process (MDP)* is leveraged to describe the problem setup. An MDP is represented by a quintuple $(\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P}, \gamma)$, which collects a state space $\mathcal{S}$; an action space $\mathcal{A}$; a scalar reward function $r_t = \mathcal{R}(\mathbf{s}_{t-1}, \mathbf{a}_{t-1}, \mathbf{s}_t) : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$, where $\mathbf{a}_t \in \mathcal{A}$ and $\mathbf{s}_t \in \mathcal{S}$; a state transition probability distribution $\mathcal{P}(\mathbf{s}_{t+1} | \mathbf{s}_t, \mathbf{a}_t) : \mathcal{S} \times \mathcal{S} \times \mathcal{A} \to [0, 1]$; and a discount factor $\gamma \in [0, 1]$. Under this setting, using experiences (i.e., control trajectories $\tau = \{\mathbf{s}_0, \mathbf{a}_0, r_1, \mathbf{s}_1, \mathbf{a}_1, ...\}$) collected via repeated interaction, an RL agent is to learn an optimal control policy, namely a state-action mapping $\pi(\mathbf{a}_t | \mathbf{s}_t)$*, that maximizes the *expected episodic discounted reward* $\mathbb{E}_{\mathbf{a}_t \sim \pi}(\sum_{t \in \mathcal{T}} \gamma^t r_t)$. Adding a discount factor γ in this reward expectation prioritizes rewards in the near future over those collected at a much later time, as the weight γ^t decays exponentially through time.

To solve (1) using RL, we define the quintuple as follows: Among $(\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P}, \gamma)$, $\mathcal{P}$ is implicitly given by the overall system dynamics (i.e., $\mathcal{F}$ in (1)) and $\gamma = 0.99$ is commonly used, which leave the remaining three to be explicitly defined. First, the continuous action space defined in Section II-A is leveraged: $\mathbf{a}_t = [\dot{m}_t^1, \dot{m}_t^2, ..., \dot{m}_t^N, T_t^{da}] \in \mathcal{A} \subset \mathbb{R}^{N+1}$. Second, to solve (1) equivalently using RL, the reward function is defined as the negative value of the single step cost: $r_t = -\mathbf{w}_t^\top \mathbf{C}_t$. Finally, the state ($\mathbf{s}_t \in \mathcal{S}$) represents the information the RL agent requires to make decisions at every step. The design of $\mathcal{S}$ essentially determines the RL controller's inputs, and is thus discussed in more detail next.

B. RL Controllers with Different State Representations

To properly guide the RL controller in decision-making, the state representation typically contains information regarding the current system status and other information related to its future evolving trajectory. As a result, we define a general form state $\mathbf{s}_t := [\mathbf{T}_t, \mathbf{T}_t^{out}, \mathbf{G}_t^{int}, \mathbf{G}_t^{solar}, \mathbf{u}_t, t, \mathbf{w}_t, \delta_t]$ in this study, with a slight abuse of notation†. Specifically,

*We use the notation $\pi(\mathbf{a}|\mathbf{s})$ instead of $\mathbf{a} = \pi(\mathbf{s})$ to follow the convention in the RL community. It indicates what action to choose at state $\mathbf{s}$ under policy π.

†Strictly, the notation of state should be written as $\mathbf{s}_t := [(\mathbf{T}_t)^\top, (\mathbf{T}_t^{out})^\top, (\mathbf{G}_t^{int})^\top, (\mathbf{G}_t^{solar})^\top, (\mathbf{u}_t)^\top, t, (\mathbf{w}_t)^\top, \delta_t]^\top$

zone temperatures $\mathbf{T}_t \in \mathbb{R}^N$, step index $t \in \mathbb{R}$, multi-objective weights $\mathbf{w}_t \in \mathbb{R}^3$ and power consumption margin $\delta_t := \overline{P}_t - P(\mathbf{a}_t, T_t^{out}) \in \mathbb{R}$ reflect the system status at current step; and a multi-step outdoor temperature vector $\mathbf{T}_t^{out} \in \mathbb{R}^K$, a flattened multi-step multi-zone internal heat gain vector $\mathbf{G}_t^{int} \in \mathbb{R}^{(N\times K)}$, a flattened multi-step multi-zone solar heat gain vector $\mathbf{G}_t^{solar} \in \mathbb{R}^{(N\times K)}$ and a DR signal vector $\mathbf{u}_t := [\overline{P}_t, \overline{P}_{t+1}, ..., \overline{P}_{t+K-1}]^\top \in \mathbb{R}^K$ reflect the factors affecting the future trajectory, where K is the number of steps considered.

Though some studies, e.g., [26], assume the exogenous data (i.e., $\mathbf{G}_t^{int}$ and $\mathbf{G}_t^{solar}$) can be predicted and used as controller inputs, we avoid using forecasts because: *a)* forecast modules increase the implementation cost; *b)* the controller's performance will be dependent on the quality of prediction; and *c)* in contrast to MPC, the RL controller does not require explicit forecasts to make decisions, as it can directly learn the mapping between historical data and the optimal action (One example is [18], in which "the last 4 frames of a history" are used to show the historical trajectory instead of future prediction).

Thus, in this study, the following three $\mathbf{s}_t$ structures that do not rely on forecast modules are considered.

1) *Historical Measurement, RL-HM*: Outdoor temperature and heat gain data from the past few hours are provided, i.e., $\mathbf{T}_{t,-K}^{out} := [T_{t-K+1}^{out}, ..., T_t^{out}]^\top \in \mathbb{R}^K$ and $\mathbf{G}_{t,-K}^{h} := [(\mathbf{G}_{t-K+1}^{h,\mathcal{N}})^\top, ..., (\mathbf{G}_t^{h,\mathcal{N}})^\top]^\top \in \mathbb{R}^{(N\times K)}$, in which $h \in \{solar, int\}$ and $\mathbf{G}_t^{h,\mathcal{N}} \in \mathbb{R}^N$ is the solar/internal heat gain at step t for all N zones. This state representation uses only historical data and does not rely on any forecasting module.

2) *Coarsely Approximated, RL-CA*: Heat gains, however, are not easily measurable without a sophisticated sensor network; therefore, we investigate using the time of day and a Boolean weekday indicator (ω) to coarsely approximate $\mathbf{G}^{int}$ and $\mathbf{G}^{solar}$. Specifically, as time of day is cyclic ordinal (e.g., outdoor environment at 1:00 is similar to that at 23:00, though 1 and 23 are apart), trigonometric encoding [42, Equation (6)] are used: $[sin_t, cos_t] = [sin(\frac{hour(t)*2\pi}{24}), cos(\frac{hour(t)*2\pi}{24})]$, in which $hour(t)$ converts the control step t to time of day in 24-hour format, to represent the temporal proximity. This state representation further removes the requirement of some sophisticated sensors.

3) *No Heat gain, RL-NH*: a state representation in which the heat gain information is completely discarded is introduced for comparison.

Table I summarizes these RL controllers with different $\mathbf{s}_t$ structures, i.e., different RL controller inputs. Column "$|\mathbf{s}_t^*|$" in the table shows the state dimension based on $K = 48$, which represents a 4-hour look back window with 5-minute control interval and will be used in the rest of the paper. Performance based on these controllers are later compared in Section V.

TABLE I
RL CONTROLLERS WITH DIFFERENT STATE REPRESENTATIONS

Controller	State Representation	$\lvert\mathbf{s}_t^*\rvert$
RL-HM	$\mathbf{s}_t^{HM} := [\mathbf{T}, \mathbf{T}_{t,-K}^{out}, \mathbf{G}_{t,-K}^{int}, \mathbf{G}_{t,-K}^{solar}, \mathbf{u}_t, t, \mathbf{w}_t, \delta_t]$	586
RL-CA	$\mathbf{s}_t^{CA} := [\mathbf{T}, \mathbf{T}_{t,-K}^{out}, \omega, \sin_t, \cos_t, \mathbf{u}_t, t, \mathbf{w}_t, \delta_t]$	109
RL-NH	$\mathbf{s}_t^{NH} := [\mathbf{T}, \mathbf{T}_{t,-K}^{out}, \mathbf{u}_t, t, \mathbf{w}_t, \delta_t]$	106

IV. TWO-STAGE RL POLICY SEARCH

In deep RL, the control policy, as a mapping relationship, is instantiated using a deep neural network, denoted by $\pi_{\boldsymbol{\theta}}(\mathbf{a}_t|\mathbf{s}_t)$, where vector $\boldsymbol{\theta}$ collects neural network parameters (i.e., weights and biases). To maximize $\mathbb{E}_{\mathbf{a}_t\sim\pi_{\boldsymbol{\theta}}}(\sum_{t\in\mathcal{T}}\gamma^t r_t)$, referred to as $J(\boldsymbol{\theta})$ hereinafter as it is a function of $\boldsymbol{\theta}$, a category of RL algorithms, called *policy gradient* methods, uses gradient ascent to update the policy iteratively by:

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \alpha\widehat{\nabla}_{\boldsymbol{\theta}}J(\boldsymbol{\theta}), \tag{6}$$

where α is the learning rate, $\widehat{\nabla}_{\boldsymbol{\theta}}J(\boldsymbol{\theta}) \approx \nabla_{\boldsymbol{\theta}}L(\boldsymbol{\theta})$ estimates the policy gradient, and $\nabla_{\boldsymbol{\theta}}L(\boldsymbol{\theta})$ is the gradient of a surrogate objective $L(\boldsymbol{\theta})$ (the construction of $L(\boldsymbol{\theta})$ varies by algorithms) that is computed via back-propagation (BP).

A. ES-RL Algorithm

In contrast to many policy gradient methods, which calculate $\widehat{\nabla}_{\boldsymbol{\theta}}J(\boldsymbol{\theta})$ using the *policy gradient theorem* via BP [17, Chapter 13], the ES-RL algorithm of [34] obtains the gradient using zero-order gradient estimation (ZOE) [35]. As proposed in [34], policy parameters $\boldsymbol{\theta}$ follows an isotropic multivariate Gaussian distribution with fixed covariance, i.e., $\boldsymbol{\theta} \sim N(\hat{\boldsymbol{\theta}}, \sigma^2\mathbf{I})$, where $\hat{\boldsymbol{\theta}}$ is the mean parameter vector to be learned and σ is a standard deviation, as a hyper-parameter. ES-RL aims at maximizing a Gaussian smoothed version of the original objective:

$$V(\hat{\boldsymbol{\theta}}) := \mathbb{E}_{\theta\sim N(\hat{\boldsymbol{\theta}},\sigma^2 I)}J(\boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{\epsilon}\sim N(0,I)}J(\hat{\boldsymbol{\theta}} + \sigma\boldsymbol{\epsilon}),$$

where the smoothness is controlled by σ. The goal of Gaussian smoothing is to introduce better properties than the original function, see [35, Section 2] for more discussion.

To estimate the gradient $V(\hat{\boldsymbol{\theta}})$, the following lemma [34], [35] is leveraged, and we provide its proof below.

Lemma 1. *The gradient of $V(\hat{\boldsymbol{\theta}})$ is estimated by*

$$\nabla V(\hat{\boldsymbol{\theta}}) \approx \frac{1}{\sigma}\mathbb{E}_{\boldsymbol{\epsilon}\sim N(0,I)}[\boldsymbol{\epsilon}\cdot J(\hat{\boldsymbol{\theta}} + \sigma\boldsymbol{\epsilon})].$$

Proof: The Taylor series expansion of $J(\hat{\boldsymbol{\theta}} + \sigma\boldsymbol{\epsilon})$ is:

$$J(\hat{\boldsymbol{\theta}} + \sigma\boldsymbol{\epsilon}) = J(\hat{\boldsymbol{\theta}}) + \sigma\boldsymbol{\epsilon}^\top\nabla J(\hat{\boldsymbol{\theta}}) + O(\sigma^2). \tag{7}$$

By taking the expectation on both sides, there is:

$$\mathbb{E}_{\boldsymbol{\epsilon}}[J(\hat{\boldsymbol{\theta}} + \sigma\boldsymbol{\epsilon})] \approx \mathbb{E}_{\boldsymbol{\epsilon}}[J(\hat{\boldsymbol{\theta}}) + \sigma\boldsymbol{\epsilon}^\top\nabla_\epsilon J(\hat{\boldsymbol{\theta}})] = J(\hat{\boldsymbol{\theta}}). \tag{8}$$

So the computation of $\nabla\mathbb{E}_{\boldsymbol{\epsilon}}[J(\hat{\boldsymbol{\theta}} + \sigma\boldsymbol{\epsilon})]$ can be approximated by $\nabla J(\hat{\boldsymbol{\theta}})$. To compute $\nabla J(\hat{\boldsymbol{\theta}})$, we multiply both sides of (7) with $\boldsymbol{\epsilon}$ and take the expectation:

$$\begin{aligned}\mathbb{E}_{\boldsymbol{\epsilon}}[\boldsymbol{\epsilon}J(\hat{\boldsymbol{\theta}} + \sigma\boldsymbol{\epsilon})] &\approx \mathbb{E}_{\boldsymbol{\epsilon}}[\boldsymbol{\epsilon}J(\hat{\boldsymbol{\theta}})] + \mathbb{E}_{\boldsymbol{\epsilon}}[\sigma\boldsymbol{\epsilon}\boldsymbol{\epsilon}^\top\nabla J(\hat{\boldsymbol{\theta}})] \\ &= \sigma\mathbb{E}_{\boldsymbol{\epsilon}}[\boldsymbol{\epsilon}\boldsymbol{\epsilon}^\top]\nabla J(\hat{\boldsymbol{\theta}}) = \sigma\nabla_\theta J(\hat{\boldsymbol{\theta}}).\end{aligned} \tag{9}$$

Equation (9) suggests $\nabla J(\hat{\boldsymbol{\theta}}) \approx \frac{1}{\sigma}\mathbb{E}_{\boldsymbol{\epsilon}}[\boldsymbol{\epsilon}J(\hat{\boldsymbol{\theta}} + \sigma\boldsymbol{\epsilon})]$. Therefore, there is $\nabla V(\hat{\boldsymbol{\theta}}) \approx \frac{1}{\sigma}\mathbb{E}_{\boldsymbol{\epsilon}\sim N(0,I)}[\boldsymbol{\epsilon}\cdot J(\hat{\boldsymbol{\theta}} + \sigma\boldsymbol{\epsilon})]$. ■

TABLE II
ZOE TRIAL PARAMETERS

Type	σ	Trial (Initial Points)
Under Smoothed	0.5	us_1 (0.0, -2.0), us_2 (6.6, -4.0)
Properly Smoothed	1.285	ps_1 (0.0, 2.0), ps_2 (6.6, 4.0)
Over Smoothed	2.5	os_1 (0.5, 4.5), os_2 (5.5, 4.5)

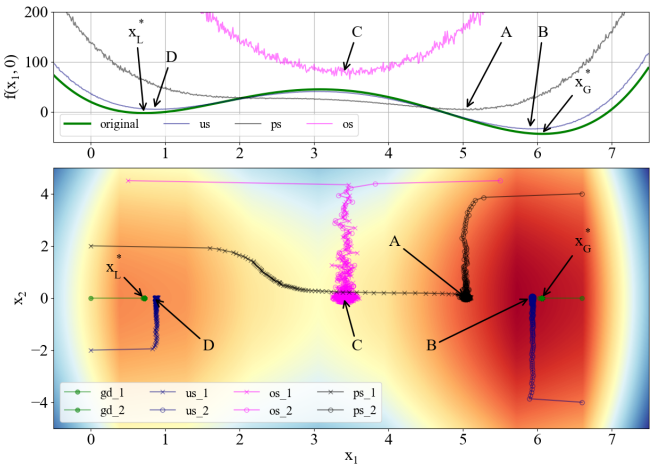


Fig. 1. Non-convex function used in this numerical example is $f(\mathbf{x}) = 20 - 70x_1 + 65.7x_1^2 - 17.1x_1^3 + 1.3x_1^4 + 1.6x_2^2$. Two standard gradient descent (gd) processes are shown to indicate $\mathbf{x}_L^*$ and $\mathbf{x}_G^*$. Top plot shows the cross-section of the original and three smoothed functions at $x_2 = 0$ and the bottom plot collects the converging trajectories. Points A/B/C/D reveal the optima of smoothed functions, which explain the trajectories in the bottom plot. Different initial points are selected for better visualization, they can be categorized into two groups depending on their proximity to $\mathbf{x}_L^*$ and $\mathbf{x}_G^*$.

This means $\nabla V(\hat{\boldsymbol{\theta}})$ can be estimated using zero-order sampled rewards $J(\hat{\boldsymbol{\theta}} + \sigma\boldsymbol{\epsilon})$, and the policy parameter vector $\hat{\boldsymbol{\theta}}$ can thus be updated. Such ZOE-based methods present the following three major advantages:

1) *BP-free*: This gradient-free feature avoids conducting computationally intensive BP and reduces the overall computation by two thirds [34].
2) *Scalability*: Though evidences show that this approach requires more iterations to converge [35], ES-RL, by utilizing shared random seeds to reduce communication requirement, can be scaled to a large number of parallel rollout workers and ultimately yields an overall shorter learning wall-time in practice.
3) *Poor policy avoidance*: By optimizing on the Gaussian smoothed objective $V(\hat{\boldsymbol{\theta}})$, such methods can potentially escape poor-performing local optima in the non-convex problem setting, which is discussed next in Section IV-B.

B. Convergence Features of ZOE based Methods

In this section, through a simple numerical experiment, two key features regarding the convergence of ZOE-based methods, including ES-RL, are illustrated. Specifically, for visualizing in $\mathbb{R}^2$, we construct an arbitrary non-convex function $f : \mathbb{R}^2 \to \mathbb{R}$ with a local and a global optimum ($\mathbf{x}_L^*$ and $\mathbf{x}_G^*$). In this experiment, ZOE-based gradient descent is used to search for a solution of $\min f(\mathbf{x})$. Specifically,

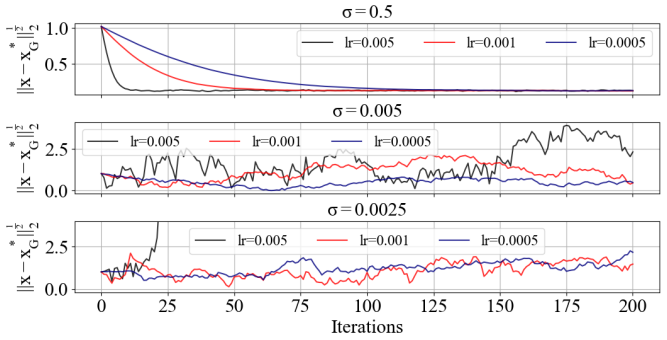


Fig. 2. Starting from 'A' in Fig. 1, in Stage II, smaller σ are used to fine-tune the solution (i.e., $A \to \mathbf{x}_G^*$). Note if σ is too small, the solution x does not converge towards $\mathbf{x}_G^*$ due to the estimation error magnified by $1/\sigma$. All curves start from $||\mathbf{x}_A - \mathbf{x}_G^*||_2^{\frac{1}{2}}$ and 'lr' indicates learning rates.

$\nabla \mathbb{E}_{\boldsymbol{\epsilon}\sim N(0,I)} f(\mathbf{x} + \sigma\boldsymbol{\epsilon}) \approx \frac{1}{\sigma}\mathbb{E}_{\boldsymbol{\epsilon}\sim N(0,I)}[\boldsymbol{\epsilon} \cdot \tilde{f}(\mathbf{x} + \sigma\boldsymbol{\epsilon})]$ is used to estimate the gradient around $\mathbf{x}$ and $\tilde{f} = f + \nu$ represent samples with noise (assume $\nu \sim \mathcal{N}(0, \sigma_\nu)$). As shown in Table II, different smoothing levels and initial points are used in six trials and the converging processes are illustrated in Fig. 1. It can be seen that, if σ is relatively small (under smoothed), similar to gradient descent, the two trajectories (i.e., *us_1* and *us_2*) converge to the closest optimum due to the non-convexity in the resulting under-smoothed surface. However, when given an adequate σ (properly smoothed), ZOE can escape the attraction of $\mathbf{x}_L^*$ and converge towards $\mathbf{x}_G^*$ due to the smoothing effect (see *ps_1* in contrast to *gd_1* and *us_1*). On the other hand, also because of the function smoothing, a larger σ for ZOE drags the converged point away from the "true optimum" $\mathbf{x}_G^*$ (see *os_1* and *os_2*). This is also true for the properly smoothed cases, which lead to a convergence to the neighborhood of $\mathbf{x}_G^*$ (point A in Fig. 1), instead of $\mathbf{x}_G^*$ itself.

To summarize, from this simple illustrative example, two key observations of the ZOE-based method's convergence features can be made:

1) An adequately large σ is needed to search with a more "global" vision and avoid poor-performing local optima;
2) Conversely, it requires σ to be small to converge "locally" to a better local optimum.

Apparently, these two requirements are conflicting.

C. Two-Stage Global-Local Policy Search in RL

To tackle the conflict for choosing σ in ZOE, one idea is to start with a larger σ and gradually reduce its value as learning progresses. However, the design of the shrinking schedule depends on the optimizing surfaces of specific problems and is out of the scope of this paper; for simplicity and generalizability, we propose to use a simple two-stage schedule instead:

Stage I (Global Policy Search): Leveraging several different smoothing levels (i.e., $\sigma^1 < \sigma^2 < ... < \sigma^k$), and initiating k different trials of ES-RL global policy search. The reason for using different σ values is because its optimal value, which properly smooths the function, is problem specific and is hard to derive theoretically. As shown in the previous

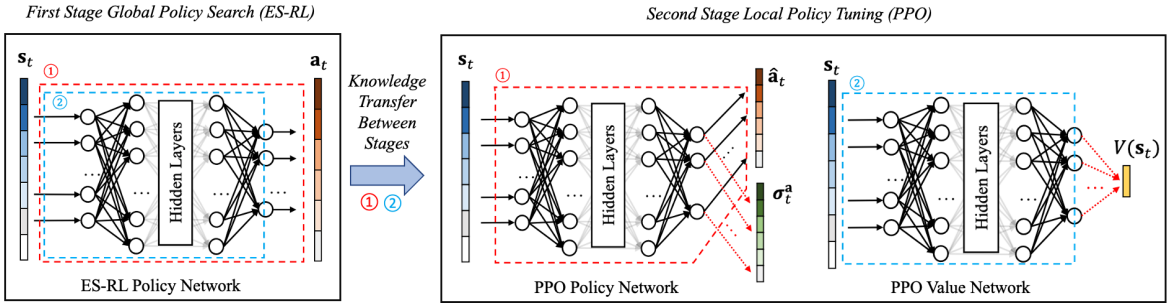


Fig. 3. Knowledge transfer between stages is implemented by network parameters copying, which replicates ES-RL-S1's behavior to Stage II. Two points worth noting: 1) Besides copying parameters for action output (see red dashed box/polygons with ①), the PPO policy network also outputs standard deviation $\sigma_t^{\mathbf{a}}$, as PPO policy is stochastic [§]($\mathbf{a}_t = N(\widehat{\mathbf{a}}_t, (\sigma_t^{\mathbf{a}})^2)$). So parameters in the last layer for generating $\sigma_t^{\mathbf{a}}$ are manually set as they do not exist in ES-RL-S1's policy network. 2) PPO requires a value network, which also does not exist in Stage I. We warm-start such a network using all the ES-RL policy network's parameters, except for those in the last layer (see blue dashed boxes marked by ②). Red dotted arrows in the figure represent parameters that do not exist in ES-RL policy network.

numerical example and later in the case study (Table IV), gradually increased σ values will first in general improve the objective function value (i.e., from "under smoothed" to "properly smoothed") and then worsen it (i.e., from "properly smoothed" to "over smoothed"), yielding a generally "U-shape" performance curve. Those σ^i with the best objective function values are those that properly smoothed the original problem, and the controllers trained with these values should enter the next, "fine-tuning", stage. These learned controllers are collectively referred to as *ES-RL-S1* hereinafter.

Stage II (Local Policy Tuning): Starting with good-performing ES-RL-S1 policies, fine-tune them until converged to the "true optimum" (move from the optimal solution of the Gaussian smoothed problem to that of the original problem, e.g., $A \rightarrow \mathbf{x}_G^*$ in Fig. 1). Concretely, there are two options for local policy fine-tuning:

a) Use ES-RL with a smaller σ: As illustrated by "us_1/us_2" in Fig. 1, if σ is small, ES-RL optima (Point D and B) are close to the optima of the original problem ($\mathbf{x}_L^*$ and $\mathbf{x}_G^*$). From this perspective, the Stage II solution can converge more closely to the "true optimum" if smaller σ is used.

b) Use a policy gradient algorithm: Since the advantages of ZOE-based methods (i.e., scalability and poor local optima avoidance) are no longer needed in Stage II, a first-order BP based method can be used, which directly optimizes the original function and leads to a more accurate convergence.

For Option a), Fig. 2 shows the example of the Stage II fine-tuning using three new smaller σ. Note, when $\sigma = 0.5$, $\mathbf{x}$ is tuned from 'A' to 'B' (not $\mathbf{x}_G^*$), but if more improvement is desired, using smaller σ (i.e., $\sigma = 0.005$ or $\sigma = 0.0025$) does not help as $||\mathbf{x} - \mathbf{x}_G^*||_2^{\frac{1}{2}}$ diverges in Fig. 2. The reason for this is that very small σ will make the gradient estimated by $\frac{1}{\sigma}\mathbb{E}_{\boldsymbol{\epsilon}}[\boldsymbol{\epsilon} J(\hat{\boldsymbol{\theta}} + \sigma\boldsymbol{\epsilon})]$ become inaccurate. As a result, in the rest of this paper, we pursue Option b), in which the *Proximal Policy Optimization (PPO)* [36] is used in Stage II. PPO is selected due to its consideration of the Kullback–Leibler (KL) divergence in learning iterations, which enables stable policy updates from an already good ES-RL-S1 controller. Please see (7) of [36] for more details on how to construct the surrogate objective function for gradient estimation. Finally, due to the change of RL algorithm, an extra process is needed to transfer knowledge between stages: parameters of the ES-RL-R1's policy network are used to initialize PPO policy networks; see Fig.3 for details of this idea. Similarly to Stage I, controllers fine-tuned in Stage II using PPO are referred to as *PPO-RL-S2* for simplicity.

Finally, it is worth noting that the ES-RL sub-optimal convergence is not discussed in the original ES-RL paper. This is possibly because most RL benchmark problems are of the task-completion type, and with the ES-RL learned sub-optimal policy, those tasks can still be completed. However, when applying RL to real-world engineering problems, e.g., cost optimization, the policy improvement can have more practical meaning and thus provides additional incentive for conducting the second stage fine-tuning.

V. Case Study

In this section, the proposed two-stage global-local policy search method is applied to train an RL controller that optimizes the multi-zone grid-interactive building control problem (1) introduced in Section II.

A. Simulation Setup

1) Multi-zone Commercial Building

A benchmark five-zone office building [43] is used ($\mathcal{N} = \{1, 2, ..., 5\}$) and key parameters are shown in Table III. The control objective is to minimize the daily weighted cost given by (1) with 5-minute control interval, which gives a control horizon/episode of 288 steps, i.e., $\mathcal{T} = \{1, 2, ..., 288\}$. Historical weather data in Austin, TX from EnergyPlus [44] are used to model the environmental disturbances. The data from July are used for training: at the beginning of an episode (one simulated day), the simulator randomly selects one of the 31 scenarios in the training data to initialize environmental disturbances, and the RL agent will learn to control the building. To examine the generalizability of the trained RL controller, data from the first ten days in August are used as testing scenarios to examine the trained policy's performance under unseen scenarios.

[§]See [34, Section 3] for the difference between smoothing in action space (e.g. PPO) and smoothing in parameter space (e.g. ES-RL).

TABLE III
PARAMETERS FOR THE FIVE-ZONE OFFICE BUILDING

Operating Constraints	$\sum_{i\in\mathcal{N}} \dot{m}_t^i \leq 10.0$ (unit: kg/s) $\dot{m}_t^i \in [0.22, 2.2], i \in \{1, ..., 4\}$, $\dot{m}_t^5 \in [0.32, 3.2]$ (Core zone) $T_t^{da} \in [10.0, 16.0]$ (unit: $^\circ C$)
AC Parameters	$a = 1.0$, $b = 0.0076$ and $c = 4.8865$
Temperature Comfort Band	$[\underline{T}^i, \overline{T}^i] = [23.0, 25.0]$ (8:00 - 18:00) $[\underline{T}^i, \overline{T}^i] = [22.0, 28.0]$ (Other time)

2) Simulation of Demand Response Events

For each day, the probability a DR event will occur is assumed to be 50%, as we envision that a more frequent DR occurrence is likely and beneficial with the prevalence of intelligent building controllers. Specifically, the start time of a DR event is uniformly sampled so that it happens during 11:00 and 18:00. The event duration is given by $l_{DR} = \lfloor 24(\xi + 1) \rfloor$ steps in which $\xi \sim U(0, 1)$ (i.e., event duration is between 2 and 4 hours. $U(0, 1)$ means a uniform distribution between $[0, 1]$ and $\lfloor \cdot \rfloor$ is the floor function). Using the same sampled ξ value, the power limit during the DR event ($t \in \mathcal{T}_{DR}$) is given by $\overline{P}_t = 30 + 20\xi$ (kW), indicating that a longer DR event usually requires less load reduction to avoid severe impact on customer's comfort. The DR power limit range $\overline{P}_t \in [30, 50]$ is selected since the building peak demand during normal operation is around 60 kW. For non-DR period ($t \notin \mathcal{T}_{DR}$), there is $\overline{P}_t = 80$ kW to represent the demand charge limit. Note, the building will receive DR notification four hours before the event starts because $\mathbf{u}_t$ in $\mathbf{s}_t$ consists of $\overline{P}_t$ for the next $K = 48$ steps. Such advance notification allows the building to take some proactive actions prior to the arrival of events. Finally, weights for different objectives are adjusted in case of DR events: $\mathbf{w}_t = [0.7, 0.2, 0.1]$ for $t \notin \mathcal{T}_{DR}$ and $\mathbf{w}_t = [0.5, 0.0, 0.5]$ otherwise. This means with the reduced power limit during a DR event, energy saving is no longer considered, shifting the emphasis to both thermal discomfort and power limit following. In practice, the configurations above, e.g., the distribution of ξ and power limit range, can be set as per building's specific requirements.

B. Learning Platform and Framework

In this study, controller training is based on a distributed reinforcement learning framework [45] using the high-performance computing (HPC) system at the U.S. National Renewable Energy Laboratory, see Fig. 4. By utilizing the HPC system and parallel learning, we can analyze the efficiency and cost for training a good-performing building controller using the state-of-the-art computing system. Specifically, hidden layers of the policy network $\pi_{\boldsymbol{\theta}}(\mathbf{a}_t|\mathbf{s}_t)$ have neuron numbers of [256, 128, 128, 64, 64, 32, 16], with "tanh" as activation function. The input and output of the neural network are normalized to $[-1.0, 1.0]$, and the reward is scaled using a factor of 0.1; both normalization and reward scaling aim at helping to train the policy network. Both ES-RL and PPO can be parallelized to multiple rollout workers/learners, with each one collecting experience from its corresponding OpenAI Gym environment [46]. Note that each environment will randomly sample ξ and exogenous inputs (e.g., $\mathbf{T}^{out}$, $\mathbf{G}^{int}$ and $\mathbf{G}^{solar}$) from a training data pool (filled with environment data from July, mentioned in Section V-A) when each episode starts, so that the RL controller is exposed to a variety of DR and environmental scenarios during training and thus is able to handle different situations when trained.

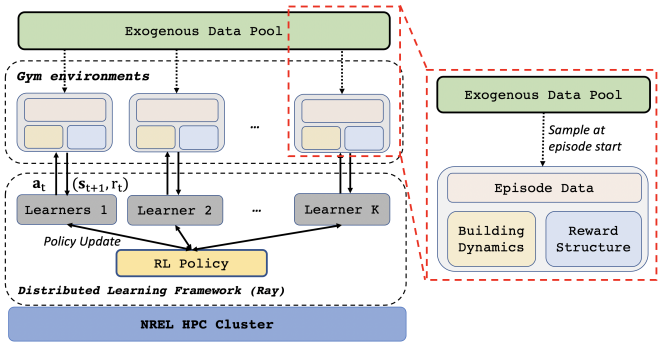


Fig. 4. A scalable reinforcement learning framework on a high-performance computing platform. For Stage I, the ES-RL learning is scaled to 20 computing nodes with a total number of 684 parallel learners. For Stage II, only one node (35 parallel learners) is used for PPO, as our experiments show scaling PPO to multiple nodes does not lead to faster convergence.

C. First Stage Global Policy Search

In this section, we show the results of global policy search using three RL inputs listed in Table I and six different objective smoothing levels. Table IV shows the average converged values ζ for each case after the RL controller is trained with ES-RL for 8×10^8 control steps. The average converged value is defined as:

$$\zeta = avr \quad \frac{1}{n_{eval}} \sum_{i=1}^{n_{eval}} c_{n_{iter}-i+1} \Bigg) \Bigg(\tag{10}$$

where $n_{eval} = 50$ is the averaging window size, n_{iter} is the number of total training iterations, c_k is the evaluated episodic cost incurred when using the RL policy trained at iteration k and the "avr" sign shows an average over multiple trials (three trials in our case). The value of ζ essentially represents the average cost a controller can achieve for the training scenarios, thus the lower value the better performance.

From Table IV, the following two observations can be made:

1) No matter which type of controller inputs, similar to the numerical experiment in Fig. 1, we observe a generally "U-shape" performance variation with respect to σ: with a small σ (e.g., $\sigma = 0.002$), the policy is trapped in local optima implied by the high ζ values, as the non-convex problem is under-smoothed; and with σ too large (e.g., $\sigma = 0.09$ or 0.15), the function smoothing effect leads the policy away from the true optima. Among all σ tested, those with medium values represent the cases of "proper-smoothing" and yield better performing policies. Though not proven analytically, this $\zeta - \sigma$ relationship observed in Table IV and Fig. 1 can help us empirically determine the best σ value to be use.

2) Regarding controller inputs, RL-CA/RL-NH perform slightly better than RL-HM in this stage, indicating the inclusion of accurate measurements of $\mathbf{G}^{int}$ and $\mathbf{G}^{solar}$ in $\mathbf{s}_t$

TABLE IV
AVERAGE CONVERGED VALUES (ζ) UNDER DIFFERENT σ

		Smoothing Standard Deviation σ					
		0.002	0.01	0.02	0.05	0.09	0.15
ζ	RL-HM	1219.96	21.08	**15.71**	16.58	34.22	49.01
	RL-CA	984.22	19.54	15.95	**15.14**	28.83	47.37
	RL-NH	865.91	19.07	16.02	**15.17**	27.61	47.41

TABLE V
SECOND STAGE POLICY TUNING IMPROVEMENT (COST REDUCTION)

σ_i	Average Episodic Cost at Policy Convergence		
	ES-RL-S1	*PPO-RL-S2*	*Improvement (%)*
0.01	18.74	14.48	22.73%
0.02	15.67	14.55	7.15%
0.05	15.09	14.17	6.49%

is not as beneficial in the office building we are investigating. One possible reason for RL-HM's slightly worse performance is because it uses a larger input dimension than RL-CA/RL-NH in the policy network, as indicated by Table I. This means it has more neural network parameters to be trained and thus requires more training iterations. Since Table IV shows the performance of all three controllers after the same number of training iterations, RL-HM having slightly worse performance is reasonable. Overall, since RL-CA does not require a sophisticated sensor network and a forecasting module, and compared with RL-NH, only three easily-accessible artificial inputs are needed, in the rest of the paper, RL-CA is used for further study.

D. Second Stage Local Policy Fine-tuning

Warm-started by the best three RL-CA ES-RL-S1 policies, denoted as π_{σ_1}, π_{σ_2} and π_{σ_3} with $\sigma_1 = 0.01$, $\sigma_2 = 0.02$ and $\sigma_3 = 0.05$ respectively, PPO is used for local policy tuning. Fig.5 shows the Stage II fine-tuning processes. It is worth mentioning that all learning curves in Fig.5 start above their corresponding dashed lines. This is because when transferring knowledge between stages, we deliberately initialize the $\sigma_t^{\mathbf{a}}$ output of the PPO policy network to encourage adequate exploration at the beginning of Stage II learning. Such exploration, inevitably, causes higher costs than those evaluated using ES-RL-S1. At the end, with all PPO learning converging to a lower cost level than their ES-RL-S1 predecessors, the benefit of the second stage tuning is manifested. Details of the Stage II performance improvement are summarized in Table V. Admittedly, in this section, the advantage of having a two-stage policy search approach is only demonstrated in the environment with training scenarios. In Section V-F, the fine-tuned PPO-RL-S2 will be examined under unseen testing scenarios to see if the performance will deteriorate (e.g., due to over-fitting the policy to the training scenarios).

E. Comparing with Single Algorithm Learning

In this section, we demonstrate the advantage of leveraging ES-RL training by comparing using PPO (the approach used in [29]) for the Stage I global policy search. Fig. 6 shows

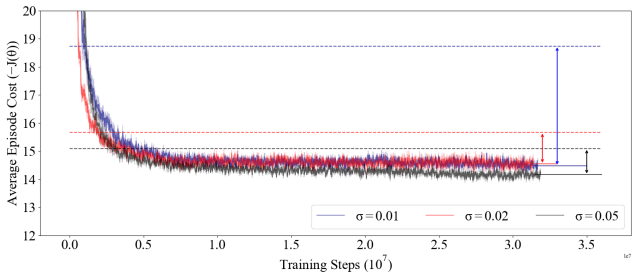


Fig. 5. Learning curves of the Stage II local policy tuning via PPO. Three solid curves depict the learning/converging processes of the Stage II policy fine-tuning, compared with the control performance of the ES-RL-S1, with costs shown by dashed lines. Each learning curve is averaged from four separate trials. Vertical double-ended arrows show the fine-tuning improvements, which correspond to the numerical values in Table V.

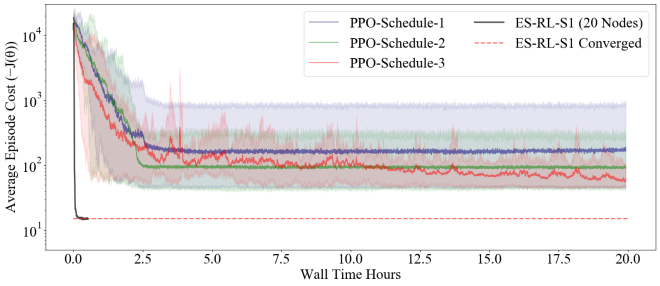


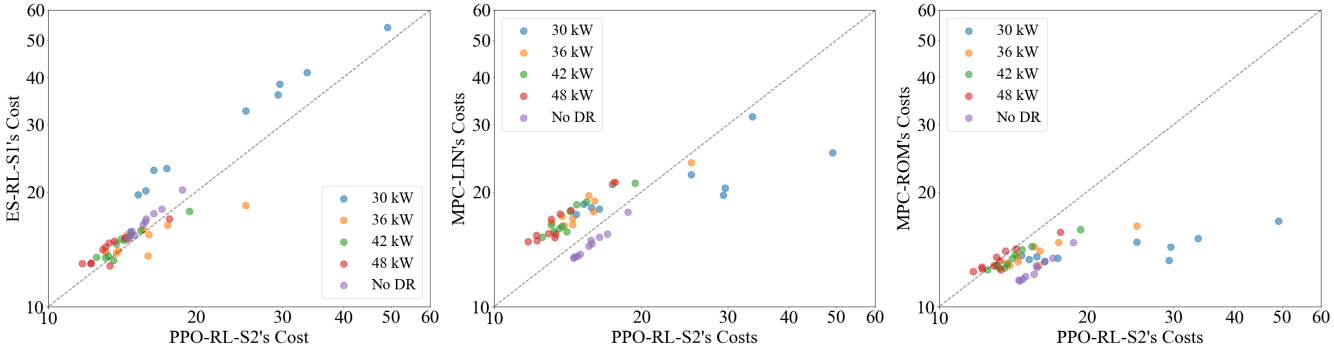
Fig. 6. Learning curves of PPO training from scratch using three learning rate schedules. Curves are averaged from multiple runs and shaded areas indicate the envelopes of all learning curves (Bottom of the envelopes show the least cost yielded by the PPO controllers). The learning curve of ES-RL-S1(σ_3) is included to compare learning performance (wall-time).

the learning curves for PPO learning from scratch with three different learning rate schedules on a single HPC node. It illustrates that after consuming $1 \times 20 = 20$ node-hours of computing resources, the best policy trained by PPO yields average cost of 46.93; in contrast, ES-RL converges to a control policy that yields an average control cost of 15.14 using $20 \times 0.5 = 10$ node-hours. The close to flat learning curves indicate that PPO learning trials are trapped in local optima. It is worth noting that we also tested PPO on multiple HPC nodes, which does not show any improvement of learning speed due to its constrained scalability (caused by synchronicity among learners and the consideration of KL divergence).

In summary, from this and the previous section, it shows the proposed two-stage learning approach demonstrates a clear advantage over either single algorithm counterpart. On the one hand, ES-RL's scalability and global searching capability are beneficial to a fast global policy search (i.e., Fig. 6); on the other hand, PPO's accurate fine-tuning can help ES-RL-S1 to further improve its performance (as demonstrated in Fig 5 and Table V). Considering both aspects, using a two-stage learning approach is beneficial.

F. Controller Performance Under Testing Scenarios

Ten days of unseen testing scenarios are used to examine whether the performance of a PPO-RL-S2 controller can be generalized.



((a)) PPO-RL-S2 vs. ES-RL-S1 ((b)) PPO-RL-S2 vs. MPC-LIN ((c)) PPO-RL-S2 vs. MPC-ROM

Fig. 7. Comparison of control performance in testing scenarios, indicated by cost, among the RL controllers and the baseline controllers. Both axes are in log scale.

1) PPO-RL-S2 vs. ES-RL-S1 in testing scenarios

First, following the previous discussion, the control performance of PPO-RL-S2 and ES-RL-S1 are compared based on testing scenarios, and the results are shown in Fig. 7 (a). Each data point represents the cost for one testing day under five different DR scenarios ($\xi = \{0.0, 0.3, 0.6, 0.9\}$ and no DR event), resulting in a total of $5 \times 10 = 50$ data points. With a majority of points lying above the diagonal line (42 out of 50), it indicates PPO-RL-S2 has better performance when compared with ES-RL-S1. This further substantiates the conclusion that the second stage local policy tuning is effective and the knowledge obtained can be generalized to unseen scenarios. Numerically, the average costs for these 50 scenarios from two controllers are 18.25 and 16.87, showing a 7.55% cost reduction.

2) PPO-RL-S2 vs. MPC Baselines

Second, PPO-RL-S2 is compared with the following two MPC baseline controllers based on the optimization formulation given by (1):

(a) *MPC-ROM* uses the exact nonlinear ARX building Reduced Order Model in (1), solving an optimization problem with non-linear equality constraints.

(b) *MPC-LIN* uses a linear approximation (through Taylor series expansion at each step) of the nonlinear building model, solving a quadratic program each step.

Both MPC controllers are implemented using the "fmincon" solver in MATLAB with the interior-point algorithm. In addition, MPC controllers require prediction values of the exogenous data in the optimization problem formulation (i.e., $\mathbf{T}_t^{out}, \mathbf{G}_t^{int}$ and $\mathbf{G}_t^{solar}, \forall t \in \mathcal{T}$). In this paper, we provide the unrealistic *perfect forecasts* to the MPC controllers, meaning the RL controller is compared with the two best performing MPC controllers. Specifically, MPC-ROM has both perfect knowledge on system dynamics and future environment, and as a result, it provides the lower bound for the possible daily cost level. Comparisons are shown in Fig. 7 (b) and (c). PPO-RL-S2 reduces average costs by 4.16% when compared with MPC-LIN and costs 25.30% more when compared with MPC-ROM (the oracle).

In summary, our results demonstrate that PPO-RL-S2 can achieve good control performance under different unseen testing scenarios, as compared with MPC-LIN using a perfect forecast.

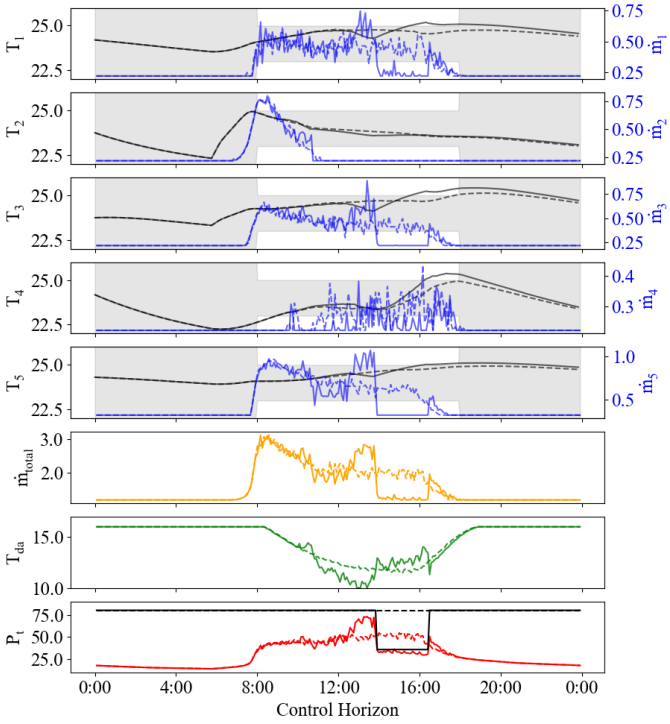


Fig. 8. Control performance of PPO-RL-S2 in one testing scenario. In all sub-figures, *dashed* lines are for the case without a DR event and the *solid* ones are for the DR scenario. Shaded areas in the first five sub-figures reveal the temperature comfort band and black lines (both dashed and solid) in the bottom figure represent the power limit ($\overline{P}_t$).

3) PPO-RL-S2 Single Scenario Control Demo

To better understand the behavior of the PPO-RL-S2 controller, we present a more detailed view of its performance on a single evaluation day, as shown in Fig.8. Two cases for the same day, one with a DR event and one without, are compared, in order to observe how the controller acts differently in case of a DR event. From Fig.8, the following observations can be made:

1) In both cases, temperatures in all zones are mostly kept in the comfort band, except for some short periods during

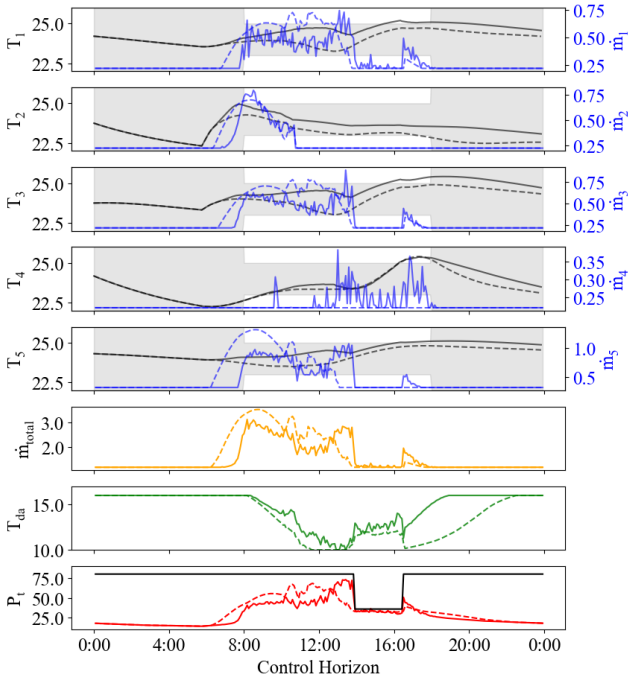


Fig. 9. Control performance of ES-RL-S1 (dashed) and PPO-RL-S2 (solid) in one testing scenario.

the DR event;

2) Power limit constraint $P_t < \overline{P}_t$ is satisfied for all $t \in \mathcal{T}$;
3) Proactive prior-event control is observed in order to prepare the building for the in-coming DR event. Specifically, demand is properly shifted: Zone 1, 3 and 5 are pre-cooled as shown by the increased $\dot{m}_i, i \in \{1, 3, 5\}$ and decreased T_{da} before the DR event, providing a larger thermal comfort margin.
4) Though not explicitly instructed, the RL controller learned from its training experience that Zone 2 (an east-facing zone) is less sensitive to the afternoon heat and thus (almost) no pre-cooling is conducted for Zone 2.
5) During the DR event, all cooling air goes to Zone 4 (the west-facing zone) to counter the thermal discomfort and $\dot{m}_i, i \in \{1, 2, 3, 5\}$ are reduced to the minimum. By doing this, P_t is successfully reduced below $\overline{P}_t$.

Note, though in this study we use a simple power limit signal, a more advanced signal from the utility, which is out of the scope of this paper, can be implemented to avoid the pre-event surge and post-event demand rebound (e.g., by gradually changing $\overline{P}_t$ prior to and after the event).

Additionally, we also compared the control behavior of ES-RL-S1 and PPO-RL-S2 controller under the single test scenario, as shown in Fig. 9. The result shows, after the Stage II fine-tuning, the RL controller can maintain the building thermal comfort and grid service quality, while using less energy. Overall, comparing with its Stage I predecessor, PPO-RL-S2 reduces the cost by 8.25%.

To summarize, this detailed scenario demonstration shows that PPO-RL-S2 has properly learned a near-optimal control behavior during training, and can provide satisfactory building control performance when deployed.

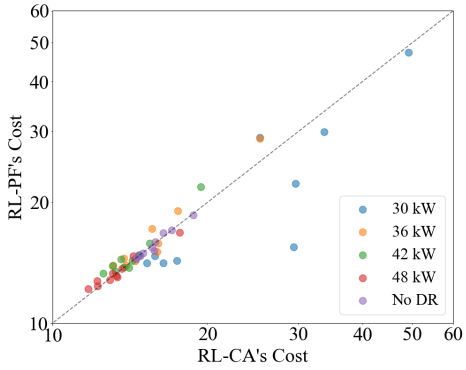


Fig. 10. Comparison of control performance in testing scenarios between RL controller with "RL-CA" and "RL-PF" state representations. "RL-CA" here is the same trained controller denoted by "PPO-RL-S2" in earlier discussion.

G. Using Forecasts in RL State Representation

So far, RL controllers that do not rely on forecast modules have been tested. In this section, we take a step back and investigate to what extent the RL controller would suffer a performance loss if forecasts were available but it did not use them.

Specifically, we introduce an RL controller "RL-PF" that uses forecasts of environment variables as part of its state representation, which is denoted as:

$$\mathbf{s}_t^{PF} := [\mathbf{T}, \mathbf{T}_{t,K}^{out}, \mathbf{G}_{t,K}^{int}, \mathbf{G}_{t,K}^{solar}, \mathbf{u}_t, t, \mathbf{w}_t, \delta_t]$$

Here, $\mathbf{T}_{t,K}^{out} := [T_t^{out}, ..., T_{t+K-1}^{out}]^\top$ and $\mathbf{G}_{t,K}^{h} := [(\mathbf{G}_t^{h,\mathcal{N}})^\top, ..., (\mathbf{G}_{t+K-1}^{h,\mathcal{N}})^\top]^\top$ for $h \in \{solar, int\}$ are perfect forecasts of environmental disturbances' future profile.

Also trained via the two-stage policy search, the "RL-PF" controller is compared with the "RL-CA" controller under various testing scenarios, see Fig. 10. The results show that, numerically, the average costs under these 50 testing scenarios are 16.49 and 16.87 for "RL-PF" and "RL-CA" respectively. This means without using perfect forecasts of disturbances, the controller performance does not deteriorate significantly. Possible reasons for this are two-fold: 1) as discussed in Section III-B, the RL controller can learn from training experiences to implicitly predict future system trajectory given historical profiles; 2) the schedule of the office building and the weather profile, though different every day, follows a regular pattern, which makes it easier for the RL controller to learn.

H. Computational Efficiency

Table VI shows the computational resources (i.e., HPC nodes and RL parallel workers) used for controller training in this study. Each computing node in the HPC system has dual 18-core processors. In Stage I, since ES-RL is highly scalable, we test learning the controller with 5, 10 and 20 nodes respectively and show that an ES-RL-S1 controller can be trained as soon as half an hour. In Stage II, we observed that PPO training performance was not improved by naïvely scaling to multiple nodes as effective scaling is bounded by a critical training batch size discussed in [47]. So PPO training is limited training to a single node using 35 parallel workers.

TABLE VI
LEARNING EFFICIENCY ON AN HPC SYSTEM

	Stage I			Stage II
HPC Node Number	5	10	20	1
RL Rollout Worker Number	144	324	684	35
Time (Hour)	1.53	0.87	0.50	6.00

As a result, Table VI shows that the fine-tuning process (as illustrated by Fig. 5) requires six hours. Note, the training time can further be reduced using GPU acceleration and/or other distributed implementations of PPO [48]. In real-life applications, equivalent computational resources from major cloud service providers are also available and spot instances, which run when the price is low, can be leveraged to reduce training costs for an RL controller.

VI. CONCLUSION AND FUTURE WORKS

This paper developed a two-stage RL policy search method to facilitate a faster convergence towards a better policy: in the global search stage, a ZOE-based RL algorithm is scaled on parallel learners to optimize a policy via a Gaussian smoothed objective function; such a policy, which ends up in the vicinity of a good-performing local optimum, will then be fine-tuned in the second stage, further improving its performance. The effectiveness of the proposed method and its advantages over existing practices were demonstrated. Though we focus on a multi-zone grid-interactive building control problem here, the proposed two-stage learning is general and can be effective in training RL policies for other control problems. Like many other data-driven methods, it is worth mentioning that the RL controller's performance might deteriorate for scenarios that are out of the distribution of the training scenarios. To cope with this, for the building control problem, we envision that the RL controller will be re-trained regularly, provided with updated training scenarios to adapt to the changes in the building and the environment.

Future studies that can bring this work closer to practice include 1) conducting experiments in additional building types other than an office building; 2) instead of a stacked state representation and a fully-connected neural network, exploring if more complicated state representations and networks (i.e., convolutional/recurrent neural network, which better handle the correlation among exogenous inputs) can provide benefits; 3) Discussing best practice for choosing training scenarios: e.g., whether the exogenous inputs include data from the previous month, or the previous two months or from the same time last year.

REFERENCES

[1] S. Somasundaram *et al.*, "Reference guide for a transaction-based building controls framework," *Pacific Northwest National Laboratory*, 2014.
[2] F. Rahimi and A. Ipakchi, "Using a transactive energy framework: Providing grid services from smart buildings," *IEEE Electrification Magazine*, vol. 4, no. 4, pp. 23–29, Dec. 2016.
[3] R. Tang and S. Wang, "Model predictive control for thermal energy storage and thermal comfort optimization of building demand response in smart grids," *Applied Energy*, vol. 242, pp. 873–882, May 2019.
[4] H. Hao, Y. Lin, A. S. Kowli, P. Barooah, and S. Meyn, "Ancillary service to the grid through control of fans in commercial building HVAC systems," *IEEE Transactions on smart grid*, vol. 5, no. 4, pp. 2066–2074, Jul. 2014.
[5] S. C. Dhulipala, R. V. A. Monteiro, R. F. da Silva Teixeira, C. Ruben, A. S. Bretas, and G. C. Guimarães, "Distributed model-predictive control strategy for distribution network volt/var control: A smart-building-based approach," *IEEE Transactions on Industry Applications*, vol. 55, no. 6, pp. 7041–7051, Sep. 2019.
[6] U.S. Energy Information Administration, "Commercial Buildings Energy Consumption Survey (CBECS) Table E3. Electricity consumption (Btu) by end use, 2012," https://www.eia.gov/consumption/commercial/data/2012/ce/cfm/e3.php, accessed: Aug. 15th, 2021.
[7] San Diego Gas & Electric, "AC Saver (Summer Saver)," accessed: Mar. 24th, 2021. [Online]. Available: https://www.sdge.com/residential/savings-center/rebates/your-heating-cooling-systems/summer-saver-program.
[8] M. Ostadijafari, A. Dubey, and N. Yu, "Linearized price-responsive HVAC controller for optimal scheduling of smart building loads," *IEEE Transactions on Smart Grid*, vol. 11, no. 4, pp. 3131–3145, Jan. 2020.
[9] J. Shi, N. Yu, and W. Yao, "Energy efficient building HVAC control algorithm with real-time occupancy prediction," *Energy Procedia*, vol. 111, pp. 267–276, Mar. 2017.
[10] M. Ostadijafari and A. Dubey, "Tube-based model predictive controller for building's heating ventilation and air conditioning (HVAC) system," *IEEE Systems Journal*, Sep. 2020.
[11] J. Cai and H. Zhang, "Predictive control of building thermal loads for participation in energy and regulation markets," in *2020 American Control Conference (ACC)*, Jul.1-3, 2020, pp. 4219–4224.
[12] D. Kim, J. Braun, J. Cai, and D. Fugate, "Development and experimental demonstration of a plug-and-play multiple RTU coordination control algorithm for small/medium commercial buildings," *Energy and Buildings*, vol. 107, pp. 279–293, Nov. 2015.
[13] X. Zhang, M. Pipattanasomporn, and S. Rahman, "A self-learning algorithm for coordinated control of rooftop units in small-and medium-sized commercial buildings," *Applied Energy*, vol. 205, pp. 1034–1049, Nov. 2017.
[14] A. Afram and F. Janabi-Sharifi, "Theory and applications of HVAC control systems–A review of model predictive control (MPC)," *Building and Environment*, vol. 72, pp. 343–355, Feb. 2014.
[15] J. Drgoňa *et al.*, "All you need to know about model predictive control for buildings," *Annual Reviews in Control*, vol. 50, pp. 190–232, Sep. 2020.
[16] X. Zhang, D. Biagioni, M. Cai, P. Graf, and S. Rahman, "An edge-cloud integrated solution for buildings demand response using reinforcement learning," *IEEE Transactions on Smart Grid*, vol. 12, no. 1, pp. 420–431, Jan. 2021.
[17] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.
[18] V. Mnih *et al.*, "Playing Atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.
[19] T. Johannink *et al.*, "Residual reinforcement learning for robot control," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, May 20-24, 2019, pp. 6023–6029.
[20] F. Ruelens, B. J. Claessens, S. Vandael, B. De Schutter, R. Babuška, and R. Belmans, "Residential demand response of thermostatically controlled loads using batch reinforcement learning," *IEEE Transactions on Smart Grid*, vol. 8, no. 5, pp. 2149–2159, Sep. 2017.
[21] F. Ruelens, B. J. Claessens, S. Quaiyum, B. De Schutter, R. Babuška, and R. Belmans, "Reinforcement learning applied to an electric water heater: from theory to practice," *IEEE Transactions on Smart Grid*, vol. 9, no. 4, pp. 3792–3800, Jul. 2018.
[22] D. J. Biagioni, X. Zhang, P. Graf, D. Sigler, and W. Jones, "A comparison of model-free and model predictive control for price responsive water heaters," in *Proceedings of the 1st International Workshop on Reinforcement Learning for Energy Management in Buildings & Cities*, Nov.17, 2020, pp. 29–33.
[23] H. Li, Z. Wan, and H. He, "Real-Time residential demand response," *IEEE Transactions on Smart Grid*, Sep. 2020.
[24] X. Zhang, X. Jin, C. Tripp, D. J. Biagioni, P. Graf, and H. Jiang, "Transferable reinforcement learning for smart homes," in *Proceedings of the 1st International Workshop on Reinforcement Learning for Energy Management in Buildings & Cities*, Nov. 17, 2020, pp. 43–47.
[25] E. Mocanu *et al.*, "On-line building energy optimization using deep reinforcement learning," *IEEE Transactions on Smart Grid*, vol. 10, no. 4, pp. 3698–3708, Jul. 2019.

[26] T. Wei, Y. Wang, and Q. Zhu, "Deep reinforcement learning for building HVAC control," in *Proceedings of the 54th Annual Design Automation Conference*, Jun.18, 2017, pp. 1–6.
[27] Z. Zhang, A. Chong, Y. Pan, C. Zhang, and K. P. Lam, "Whole building energy model for HVAC optimal control: A practical framework based on deep reinforcement learning," *Energy and Buildings*, vol. 199, pp. 472–490, Sep. 2019.
[28] N. S. Raman, A. M. Devraj, P. Barooah, and S. P. Meyn, "Reinforcement Learning for Control of Building HVAC Systems," in *2020 American Control Conference (ACC)*. IEEE, Jul.1-3, 2020, pp. 2326–2332.
[29] D. Azuatalam, W.-L. Lee, F. de Nijs, and A. Liebman, "Reinforcement learning for whole-building HVAC control and demand response," *Energy and AI*, vol. 2, p. 100020, Nov. 2020.
[30] M. Fazel, R. Ge, S. Kakade, and M. Mesbahi, "Global convergence of policy gradient methods for the linear quadratic regulator," in *Proceedings of the 35th International Conference on Machine Learning*, Jul. 10-15, 2018, pp. 1467–1476.
[31] Y. E. Nesterov, "A method of solving a convex programming problem with convergence rate O(1/k^2)," in *Doklady Akademii Nauk*, vol. 269, no. 3. Russian Academy of Sciences, 1983, pp. 543–547.
[32] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.
[33] X. Zhang, R. Chintala, A. Bernstein, P. Graf, and X. Jin, "Grid-interactive multi-zone building control using reinforcement learning with global-local policy search," in *2021 American Control Conference (ACC)*, May 25-28, 2021, pp. 4155–4162.
[34] T. Salimans, J. Ho, X. Chen, S. Sidor, and I. Sutskever, "Evolution strategies as a scalable alternative to reinforcement learning," *arXiv preprint arXiv:1703.03864*, 2017.
[35] Y. Nesterov and V. Spokoiny, "Random gradient-free minimization of convex functions," *Foundations of Computational Mathematics*, vol. 17, no. 2, pp. 527–566, Nov. 2017.
[36] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.
[37] J. She, "Combining PPO and evolutionary strategies for better policy search," accessed: Nov. 6th, 2021. [Online]. Available: https://cs229.stanford.edu/proj2018/report/251.pdf.
[38] M. H. Albadi and E. F. El-Saadany, "A summary of demand response in electricity markets," *Electric power systems research*, vol. 78, no. 11, pp. 1989–1996, Nov. 2008.
[39] Y. Liu *et al.*, "Coordinating the operations of smart buildings in smart grids," *Applied Energy*, vol. 228, pp. 2510–2525, Oct. 2018.
[40] J. Qin, K. Poolla, and P. Varaiya, "Mobile storage for demand charge reduction," *IEEE Transactions on Intelligent Transportation Systems*, Jun. 2021.
[41] R. H. Chintala and B. P. Rasmussen, "Automated multi-zone linear parametric black box modeling approach for building HVAC systems," in *Dynamic Systems and Control Conference*. American Society of Mechanical Engineers, Oct. 28–30, 2015.
[42] X. Zhang, M. Pipattanasomporn, T. Chen, and S. Rahman, "An IoT-based thermal model learning framework for smart buildings," *IEEE Internet of Things Journal*, vol. 199, pp. 472–490, Jan. 2020.
[43] M. Deru *et al.*, "US Department of Energy commercial reference building models of the national building stock," Tech. Rep., 2011.
[44] EnergyPlus, "Austin Weather Data," accessed: Mar. 24th, 2021. [Online]. Available: https://energyplus.net/weather-location/north_and_central_america_wmo_region_4/USA/TX/USA_TX_Austin.722540_TMY2.
[45] E. Liang *et al.*, "RLlib: Abstractions for distributed reinforcement learning," in *Proceedings of the 35th International Conference on Machine Learning*, vol. 80. PMLR, Jul. 2018, pp. 3053–3062.
[46] G. Brockman *et al.*, "OpenAI Gym," *arXiv preprint arXiv:1606.01540*, 2016.
[47] S. McCandlish, J. Kaplan, D. Amodei, and O. D. Team, "An empirical model of large-batch training," *arXiv preprint arXiv:1812.06162*, 2018.
[48] E. Wijmans *et al.*, "DD-PPO: Learning near-perfect pointgoal navigators from 2.5 billion frames," *arXiv preprint arXiv:1911.00357*, 2019.