The Thirty-Eighth AAAI Conference on Artificial Intelligence (AAAI-24)

Active Reinforcement Learning for Robust Building Control

Doseok Jang, Larry Yan, Lucas Spangher, Costas J. Spanos
University of California, Berkeley
djang@berkeley.edu, yanlarry@berkeley.edu, lucas_spangher@berkeley.edu, spanos@berkeley.edu

Abstract

Reinforcement learning (RL) is a powerful tool for optimal control that has found great success in Atari games, the game of Go, robotic control, and building optimization. RL is also very brittle; agents often overfit to their training environment and fail to generalize to new settings. Unsupervised environment design (UED) has been proposed as a solution to this problem, in which the agent trains in environments that have been specially selected to help it learn. Previous UED algorithms focus on trying to train an RL agent that generalizes across a large distribution of environments. This is not necessarily desirable when we wish to prioritize performance in one environment over others. In this work, we will be examining the setting of robust RL building control, where we wish to train an RL agent that prioritizes performing well in normal weather while still being robust to extreme weather conditions. We demonstrate a novel UED algorithm, ActivePLR, that uses uncertainty-aware neural network architectures to generate new training environments at the limit of the RL agent's ability while being able to prioritize performance in a desired base environment. We show that ActivePLR is able to outperform state-of-the-art UED algorithms in minimizing energy usage while maximizing occupant comfort in the setting of building control.

Introduction

Reinforcement learning has demonstrated remarkable success in solving sequential decision-making tasks such as the game of Go (Silver et al. 2017), Atari games (Mnih et al. 2013), energy pricing (Jang et al. 2021; Gunn et al. 2022), and many others. However, RL agents often overfit to their training environment and fail to generalize to new environments(Zhang et al. 2018). This is a serious issue in tasks where we expect underlying dynamics in the environment not to stay static, e.g. when there is distribution shift between the training environment and the test environment. Here, we explore the use of RL in residential and commercial building control, where most often an agent is trained to optimize performance in normal weather conditions. When underlying weather conditions exhibit extremes or drift due to long term effects such as climate change, control often fails. We endeavor to address distribution shift by using uncertainty to select training environments such that the resulting RL agent performs well in average conditions and is also robust to uncommon but dangerous scenarios in the test environment.

Overview of HVAC Setpoint Control

Our focus is on robust RL for building energy consumption, which represent 73% of electricity usage and 40% of greenhouse gases in the US (U.S. Department of Energy—EIA 2020). Buildings are generating increasingly large amounts of sensory information that can be used to increase energy efficiency, such as temperature, airflow, humidity, occupancy, light, and energy usage (Hayat et al. 2019).

There are several ways buildings can be automatically controlled: heating, ventilation, and air conditioning (HVAC) units, energy storage systems, plug-in electric vehicles, photovoltaic power sources, and lights (Gong et al. 2022). We will focus on HVAC as it represents roughly a third of total building energy consumption (Wemhoff and Frank 2010). Traditionally, HVAC setpoint control has been approached through model-predictive control (MPC, Kou et al. 2021) or a heuristic rule-based-controller (RBC, Mathews et al. 2001).[1] MPC is generally not scalable to high dimensional input or output spaces compared to RL, and heuristics are inflexible.

Recently, RL-based HVAC setpoint control has grown in popularity(Das et al. 2022). Rizvi and Pertzborn (2022) used Q-learning to control HVAC setpoints in the presence of unseen disturbances. Kurte et al. (2020) demonstrated how RL and meta-RL could train an RL HVAC agent that quickly adapts to different buildings. Xu et al. (2020) explored how to use transfer learning to train an RL HVAC controller that outperforms a RBC baseline across a variety of simulated buildings and climates. Figure 1 illustrates the flow of information in our RL HVAC control setup.

As climate change continues, we will experience more droughts, heat waves, rising temperatures, and cold snaps (Masson-Delmotte et al. 2021). These extreme weather events are likely underrepresented in the training data, but are essential to account for in reliable and safe building control. For example, consider an RL controller trained on the

---

[1] "setpoints" are the numbers a human might input into their thermostat to tell the HVAC systems their desired temperature. "Setpoint control" is the problem of automatically determining these setpoints to optimize some objective.

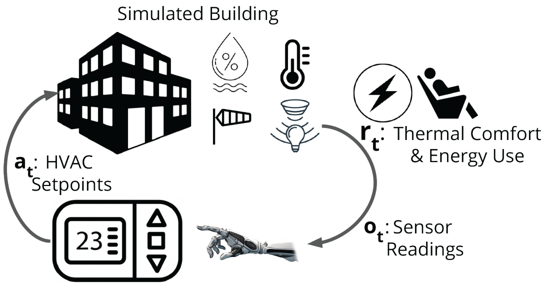

Figure 1: HVAC setpoint control in Sinergym. A simulated building sends sensor data as observations to an RL agent, which responds with HVAC setpoints as actions, and is rewarded according to energy use and thermal comfort.

dry weather of the fictional country of Desertland. If the climate of Desertland changes to have increased humidity, the RL controller may react by raising temperature during a heat wave, which is energy-inefficient, unacceptable for occupant comfort, and may even threaten occupant health.

To our knowledge, the problem of training RL HVAC controllers that are robust to changing weather is understudied, and current work focuses on detecting climate change and retraining RL controllers as their performance drops over time (Naug, Quinones-Grueiro, and Biswas 2022; Deng, Zhang, and Qi 2022), or training RL controllers that can be transferred to buildings in other climates(Xu et al. 2020; Lissa, Schukat, and Barrett 2020). These works focus on RL training pipelines that are robust to long-term changes in climate, but will still underperform during short-term extreme weather events that are rare in their training distribution.

Unsupervised Environment Design

The process of automatically selecting areas of the state space to explore is known as active learning, or as optimal experiment design. Active learning has mostly been explored for supervised learning. For example, Cohn, Atlas, and Ladner (1994); Faria et al. (2022) find regions of uncertainty in the data distribution through misclassification rates and output entropy. Makili, Sánchez, and Dormido-Canto (2012) uses conformal prediction to quantify the similarity of new data points to their dataset. EVOP (Lynch 2003) uses the sequential simplex method in order to identify experiment configurations that can maximize information gain. Bouneffouf (2016) use random exploration to identify new, promising data samples. Some of these concepts are already in use in many RL algorithms; for example, the RL algorithm we use in this paper (PPO Schulman et al. 2017), is incentivized to explore new data samples via random exploration and increasing the output action distribution entropy.

Our setting of Unsupervised Environment Design (UED, Dennis et al. 2020) is related, but different from active learning, in that we are not directly selecting training data *points* to sample, but selecting *parameters* of the environment that generates training data points. This is a more helpful problem setting in RL, as RL agents perform well with on-policy data that is collected as the RL agent explores the environment. UED is the problem of selecting new environment parameters that maximize the RL agent's generalization across diverse environments.[2] Parker-Holder et al. (2022); Dennis et al. (2020) find adversarial but feasible environment configurations with high regret; Jiang, Grefenstette, and Rocktäschel (2021) re-samples previously seen environments based on their 1-step TD error. These algorithms focus on training an RL agent that performs well across a distribution of similar tasks. SAMPLR (Jiang et al. 2022), a recent method, introduces the concept of curriculum induced covariate shift (CICS) and addresses it by launching several child simulations at each step to explore other step trajectories, but this approach does not scale computationally, especially when in an area such as building control, initializing state of the art building physics simulations comprises a large proportion of the overall computation.

Contributions

We present a novel gradient-based algorithm for UED in building control called ActivePLR that leverages agent uncertainty. To the best of our knowledge, this is the first time neural network uncertainty has been incorporated into the problem of UED; most current works focus on some form of regret; this is also the first time environment configuration variables have been directly optimized under gradient ascent rather than through some evolutionary process (Parker-Holder et al. 2022), resampling procedure (Jiang, Grefenstette, and Rocktäschel 2021), or training a separate teacher network to select new environments (Dennis et al. 2020). To the best of our knowledge, we are also the first to focus on training RL HVAC agents that are robust to short-term extreme weather events rather than long-term climate change.

We demonstrate how ActivePLR trains RL HVAC controllers that are (1) more performant overall, (2) robust to extreme weather conditions, and (3) more robust to the Sim2Real transfer than the current state-of-the-art in UED.

Methods

Reinforcement Learning (RL)

RL is a framework for finding the optimal policy in an environment. Environments are formalized by a Markov Decision Process (MDP), which consists of a tuple (S, A, T, R). State and Action spaces (S and A) consist of tuples of some fixed length indexed per timestep t; $T \colon S \times A \to S$ is a transition function; and $R \colon S \times A \times S \to \mathbb{R}$ is a reward function. Agents choose actions according to a probability distribution p_{π_θ}, determined by a policy π_θ with parameters θ to optimize the RL objective $J(\theta)$, defined $J(\theta) = \mathbb{E}_\pi \left[\sum_{s_t, a_t \sim p_\pi} [r(s_t, a_t)] \right]$. (See Sutton and Barto 2018.)

We use the PPO (Schulman et al. 2017) RL algorithm due to its performance and previous use in building control. Many works (Chen, Cai, and Bergés 2019; Zhang et al. 2021) have used PPO for HVAC control. Although we focus

---

[2] Environments that can change behavior according to configuration parameters are often referred to as Procedural Content Generation environments (Risi and Togelius 2020)

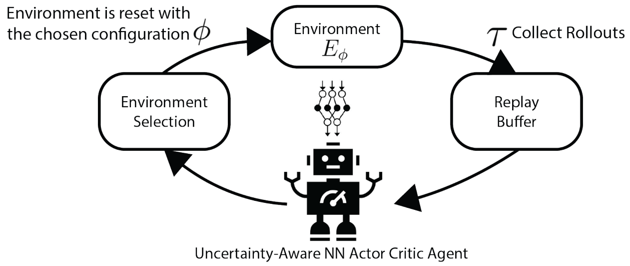

Figure 2: The flow of data during ActivePLR training.

on PPO in this paper, our algorithm should easily extend to any actor-critic RL algorithm.

Uncertainty Estimation

To estimate the uncertainty of our RL agent, we use Monte Carlo Dropout (Gal and Ghahramani 2016), in which nodes in the neural network are set to zero (“dropped out”) at random. This is used at inference time to generate multiple predictions for an individual input from different variations of the same model. The variance in these predictions is then used as a measure of the model’s uncertainty. We use Monte Carlo Dropout as opposed to other methods of estimating neural network uncertainty such as bootstrapped ensembles (Lakshminarayanan, Pritzel, and Blundell 2017) or Bayesian neural networks (Wang et al. 2020) because Monte Carlo Dropout is simpler and cheaper to train than bootstrapped ensembles and Bayesian neural networks while still providing good quantifications of uncertainty.

Formally, suppose we have a neural network $f_\theta := \mathbb{R}^n \rightarrow \mathbb{R}^m$ mapping n-dimensional input vectors to m-dimensional output vectors. f is parameterized by a list of l weight matrices $\theta := W_i|_{i=1}^{l}$, where $W_i \in \mathbb{R}^{N_{i-1}\times N_i}$ denotes the weight matrix for layer i of the neural network and N_i denotes the number of neurons in that layer. We denote the dropout operation by $d_p \colon \mathbb{R}^k \rightarrow \mathbb{R}^k$, where $d_p(\theta)$ zeros-out each column of W_i in θ with probability p. We define the uncertainty L for input $x \in \mathbb{R}^n$ as $L(x,\theta) = \mathrm{Var}(f_{d_p(\theta)}(x))$, estimated by

$$L(x,\theta) = \frac{1}{C}\sum_{c=1}^{C} f_{d_p(\theta)}(x)^T f_{d_p(\theta)}(x) - \mathbb{E}\left[f_{d_p(\theta)}(x)\right]^T \mathbb{E}\left[f_{d_p(\theta)}(x)\right] \qquad (1)$$

Essentially, we conduct C independent stochastic forward passes through the model with dropout at inference time, and use the sample variance of the outputs as our uncertainty metric. To estimate the uncertainty of our RL agent, we use the uncertainty of the critic network similar to other works (An et al. 2021; Wu et al. 2021).

Robust Prioritized Level Replay

One key assumption we make in the design of this algorithm is that, at the beginning of each training episode, we interact with an RL environment that can change its dynamics in response to some configuration parameters; for example, in this work we focus on a building simulation environment that can change its simulated weather patterns in response to weather configuration variables that we provide at the beginning of each training episode.[3] Prioritized Level Replay (PLR, Jiang, Grefenstette, and Rocktäschel 2021) is a state-of-the-art framework for selectively sampling training levels in environments with procedurally generated content. Levels with higher value loss are prioritized, inducing an emergent curriculum of increasingly difficult levels.[4] For each episode, PLR samples $d \sim P_D$, to decide whether to sample a new level from the training distribution Λ_{train} or pick one from the replay buffer Λ_{seen}. Jiang, Grefenstette, and Rocktäschel (2021) parameterized P_D as a Bernoulli distribution with probability $p = \frac{|\Lambda_{\text{seen}}|}{|\Lambda_{\text{train}}|}$. Since we consider the setting where ϕ is continuous, $|\Lambda_{\text{train}}|$ is infinite, so we set the denominator as a hyperparameter N_{PLR}, so $p = \frac{|\Lambda_{\text{seen}}|}{N_{PLR}}$.

The probability of each level in the replay buffer being sampled is determined by the value loss, and how stale that estimate of the value loss is. If we do not sample a level from the replay buffer, we sample a new one from the training distribution and add it to the buffer. In this paper, vanilla PLR samples new levels to add to the buffer uniformly at random from the set of all possible training environments. Recently, Jiang et al. (2021) proposed Robust PLR, in which only training on the PLR-selected levels, and stopping gradient updates from the randomly selected levels, generally performs better; we will refer to this variant as RPLR.

ActivePLR

We present ActivePLR: a novel addition to PLR that samples new levels to add to the replay buffer through an uncertainty-based optimization procedure instead of at random.[5] In order to generate new environments at the frontier of the agent’s uncertainty, we will backpropagate gradients from the uncertainty back to the state variable and do gradient ascent.[6] This will allow us to find the state at which the agent is most uncertain, and generate a new environment that allows the agent to interact with the world at that state. Formally, assume we have a environment E with some parameters $\phi \in \mathbb{R}^k$ that are part of the agent’s initial state space S. That is, the state $s_0 \in \mathbb{R}^n$ can be divided into ϕ and $\overline{s_0}$, where $\overline{s_0} \in \mathbb{R}^{n-k}$ is defined such that $s_0 = [\phi, \overline{s_0}]$ (and [] is the concatenation operator). We can define an objective function that tries to maximize the uncertainty of the RL agent:

$$O(\phi_i, \overline{s_0}, \theta) = L([\phi_i, \overline{s_0}], \theta) \qquad (2)$$

where L is our uncertainty estimate. We can then update ϕ:

$$\phi_{i+1} = \phi_i + \eta \nabla_{\phi_i} O(\phi_i, \overline{s_0}, \theta) \qquad (3)$$

---

[3] These environments are also known as Procedural Content Generation (PCG (Risi and Togelius 2020)) environments.

[4] In our case, a ”level” is just an environment configuration ϕ. We use the term ”level” in describing PLR to be consistent with the original paper (Jiang, Grefenstette, and Rocktäschel 2021).

[5] We also tried to use RPLR with our uncertainty-based approach, but we found it performs worse than ActivePLR

[6] We make the assumption that at least some of the environment configuration variables are continuous

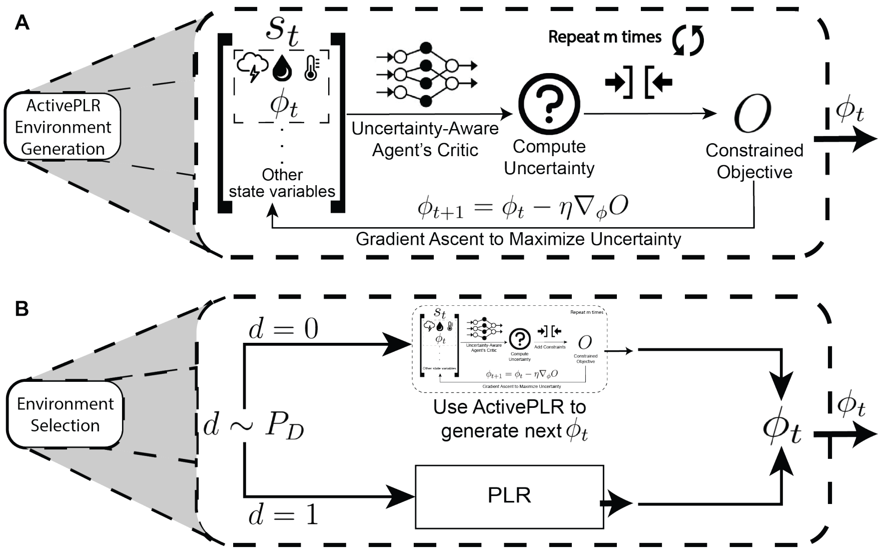

Figure 3: A. The ActivePLR environment generation process. B. The overall ActivePLR environment sampling process.

We use this optimization procedure to identify novel training environments to add to PLR's environment replay buffer. We call this procedure ActivePLR, as it is an active learning method that seeks to identify what data would be most useful for the RL agent.[7] We can also use this optimization procedure to generate all the training environments instead of using a replay buffer.[8] We denote this case as ActiveRL.

Trying to identify parameters ϕ by maximizing uncertainty can lead to unrealistic parameters that are outside of the test distribution and not useful for learning. Thus, we integrate both hard constraints and soft constraints on ϕ generation in ActivePLR. The hard constraints are useful when trying to train an RL agent that generalizes over a certain region of the ϕ space, and the soft constraints are useful when trying to train an RL agent that emphasizes performance near a particular ϕ_0, which is still robust to different values of ϕ. We will refer to the latter setting as the ϕ_0-neighborhood setting for brevity.

The hard constraints constrain the search space within some lower and upper limits specified by the user for ϕ using the extragradient (Korpelevich 1976) method. Suppose we have a lower bound constraint $\phi > b$ for some $b \in \mathbb{R}^k$ and an upper bound constraint $\phi < a$ for some $a \in \mathbb{R}^k$. Then we can use Lagrangian optimization methods from the Cooper library (Gallego-Posada and Ramirez 2022) to search for a ϕ with high uncertainty within the bounds of a and b, helping to avoid unrealistic values of ϕ.[9] Throughout this paper, we will use the implementation of ExtragradientAdam from Gallego-Posada and Ramirez (2022), which adjusts the learning rate η for each parameter according to Adam (Kingma and Ba 2014).

In the ϕ_0-neighborhood setting, the hard constraints are not enough – there is no guarantee that states near ϕ_0 will be sampled. Thus we introduce a soft constraint to O to minimize the Euclidean distance from ϕ to ϕ_0.

$$O(\phi_i, s_0, \theta) = L([\phi, \overline{s_0}], \theta) - \gamma||(\phi - \phi_0)||_2 \qquad (4)$$

where γ emphasizes the soft constraint.

Experiments

Environment

We use a modified version of the Sinergym (Jiménez-Raboso et al. 2021) OpenAI Gym (Brockman et al. 2016) environment to simulate buildings in different weather conditions. The environment uses the EnergyPlus (Crawley et al. 2001) simulation engine to model the dynamics of building systems. We use the ”5Zone” building provided with Sinergym: a $463.6m^2$ single-story building with a DX cooling coil and gas heating coils that is divided into 5 zones (1 indoor and 4 outdoor). Actions in Sinergym are continuous, two-dimensional vectors, where the agent can control the heating and cooling setpoints for the HVAC systems. We use a reward function that rewards high occupant comfort and low energy use. To quantify occupant comfort, we use the Fanger Percentage of People Dissatisfied (PPD, Fanger 1967). Energy use is the total HVAC electricity demand rate in Watts (W). The reward can be formulated as:

$$\begin{aligned} R_t = &-\rho * \lambda_E * P_t - \\ &(1-\rho) * \lambda_P * PPD_t * \mathbb{1}_{(occupancy_t > 0)} * \mathbb{1}_{PPD_t > 20} \end{aligned} \qquad (5)$$

where ρ controls how much to weight comfort against energy use, P_t is the electricity demand rate, λ_E and λ_P are scaling factors to account for varying units, $\mathbb{1}_{(occupancy_t > 0)}$ ensures there is no penalty for uncomfortable conditions when there are no occupants, and $\mathbb{1}_{PPD_t > 20}$ ensures the

---

[7] Pseudocode for ActivePLR can be found in Algorithm 1, and an illustration can be found in Figure 3.

[8] This is equivalent to ActivePLR with P_D assigning 100% probability to $d = 0$

[9] A more detailed treatment of the constrained optimization problem can be found in [12].

Algorithm 1: ActivePLR

```
procedure ActivePLR($\theta, s_0, N, T, \eta, \gamma, a, b, \rho, N_PLR$)
        ▷ $\theta$: policy parameters    ▷ $s_0$: initial state to seed environment generation    ▷ $T$: # of iterations to run PPO    ▷ $N$: # of iterations to optimize $\phi$    ▷ $\eta$: Learning rate for optimizing $\phi$    ▷ $\gamma$: Weight on soft constraint    ▷ $a$: $\phi$ lower bounds    ▷ $b$: $\phi$ upper bounds    ▷ $c$: Global episode counter    ▷ $\Lambda_{seen}$: Visited levels    ▷ $S$: Global level scores    ▷ $C$: Global level timestamps (when they were last sampled)    ▷ $\rho$: PLR staleness weighting    ▷ $N_{PLR}$: hyperparameter of $P_D$
    $\phi_0 \leftarrow$ ExtractPhi($s_0$)
    $c \leftarrow c + 1$
    for t=0 to T do
        Sample replay decision $d \sim P_D(N_{PLR})$
        if $d == 1$ then
            ScoreProb $\leftarrow P_S(\phi|\Lambda_{seen}, S)$
            StaleProb $\leftarrow P_C(\phi|\Lambda_{seen}, C, c)$
            Sample $\phi \sim (1 - \rho) \cdot$ ScoreProb $+ \rho \cdot$ StaleProb
        else
            for i=0 to N do
                $\phi \leftarrow$ ExtractPhi($s_0$)
                dist $\leftarrow ||\phi - \phi_0||_2$
                $O \leftarrow$ UncertaintyEstimate($f_\theta, s_0$) $- \gamma \cdot$ dist
                $\phi \leftarrow$ ExtragradientUpdate($\phi, O, a, b$)
                $s_0$ = Concatenate([$\phi$, $\overline{s_0}$])
            end for
        end if
        Define new index $i \leftarrow |S| + 1$
        Add $\phi_i \leftarrow \phi$ to $\Lambda_{seen}$
        Add initial value $S_i = 0$ to $S$ and $C_i = 0$ to $C$
        $\tau \leftarrow$ CollectTrajectories($E_\phi$)
        Update score $S_i \leftarrow$ PPOValueLoss($\tau, \theta$)
        Update timestamp $C_i \leftarrow c$
        $\theta \leftarrow$ PPOUpdate($\tau, \theta$)
    end for
    Return $\theta$
end procedure
```

agent is not penalized if the PPD is below the ASHRAE guidelines' comfort threshold of 20% (ANSI and ASHRAE 2017). We use $\lambda_E = 0.0001$, $\lambda_P = 0.1$, $\rho = 0.5$. The state is a continuous, 20 dimensional vector that includes 5 outdoor weather variables: outdoor air temperature, outdoor relative humidity, wind speed, wind direction, solar irradiance, and 15 other variables: indoor air temperature, indoor relative humidity, clothing value, thermal comfort, current HVAC setpoints, total HVAC electricity demand rate, occupancy count, and date (year, month, day, hour).

To simulate outdoor weather, Sinergym takes as input a file with hourly measurements of each outdoor weather variable. Originally, Sinergym added noise to outdoor temperature through an Ornstein-Uhlenbeck (OU, Doob 1942) process, to help prevent overfitting the agent to a static weather pattern. We modified Sinergym so it could add this noise to the other outdoor weather variables as well. An OU process has three parameters: σ, μ, and τ. σ controls the variance of the added noise, μ is the average value of the noise, and τ determines how quickly the noise reverts to the mean.

We can obtain reasonable values for σ and τ for each weather variable from the original input weather file, so we have 5 remaining parameters to customize Sinergym: the μ offset parameters for each weather variable.[10] Thus the ϕ for Sinergym that we vary to attempt to train a robust RL agent, is the 5 dimensional vector $< \mu_1, \mu_2, \mu_3, \mu_4, \mu_5 >$. These essentially change the average outdoor temperature, relative humidity, wind speed, wind direction, and solar irradiance over the course of the simulation. Varying the environment configuration $\phi \in \mathbb{R}^5$ enables us to collect training data from diverse outdoor weather conditions.

All experiments used 24 Intel Xeon E5-2670 CPUs.[11]

Baseline Algorithms for HVAC Control

Our basic RL baseline is a PPO agent composed of a neural network with two hidden layers, each with 256 neurons, using dropout and ReLU activations, that is trained on ϕ_0.

The most common method of training agents that generalize across diverse environments is domain randomization (DR), where ϕ is selected uniformly at random. This method is often used so agents can transfer from simulation to the real world (Chen et al. 2021; Vuong et al. 2019; Tobin et al. 2017). In the buildings domain, Jang et al. (2021) used DR to train an RL energy pricing agent to be robust to the Sim2Real transfer. In our setting: we have some lower bounds $a \in \mathbb{R}^5$ and upper bounds $b \in \mathbb{R}^5$ for each variable, described in the appendix[12]. We sample $\phi \sim U(a, b)$, where U is the uniform distribution.

Jiang et al. (2022) propose Sampled Matched PLR (SAMPLR) to generalize across diverse environments while combating the problem of curriculum-induced covariate shift (CICS), in which the distribution of training environments ($\phi \sim P$) generated through UED may become too different from the distribution of test environments ($\phi_{test} \sim \bar{P}$). Unfortunately we were not able to use SAMPLR as a baseline: SAMPLR resets the simulator at every timestep to collect fictitious trajectories; thus, SAMPLR is infeasible when a reset is expensive compared to a timestep, which is true for Sinergym and many other simulators.[13]

We use a rule-based controller (RBC) based on Sinergym's RBC, and a random controller as baselines. The random controller outputs a random cooling setpoint $a[0] \sim Uniform(22.5, 30.0)$ and a heating setpoint $a[1] \sim Uniform(15, 22.5)$.[14] Sinergym's RBC sets the desired temperature range (26-29°C) higher in the summer and lower in the winter (20-23.5°C) to minimize energy consumption. We added a rule that if no occupants are in the

---

[10] See the appendix [12] for details

[11] Code is available at https://github.com/Demosthen/ActiveRL

[12] The appendix can be found in the expanded version here: https://arxiv.org/abs/2312.10289

[13] Sinergym takes $\sim$ 3 seconds to reset to a new weather pattern. Each of our experiments involved training for 3M timesteps, where episodes were reset every 8760 timesteps. Implementing SAMPLR would have increased the time spent resetting Sinergym by 8760x, for a total of 9M seconds, or 104 days.

[14] The specific values are taken from Sinergym.

building, the RBC sets setpoints with a wide enough range that the HVAC system is turned off. We added this new occupancy-based rule for the sake of a fair comparison because we included occupancy information in the reward.

Our last RL baseline, RPLR, is described in the appendix[12], as well as hyperparameters for all methods.

Evaluating ActivePLR's Robustness to Extreme Weather Events and the Sim2Real Jump

In order to evaluate how robust agents trained by each algorithm are to extreme weather events, we evaluate each agent in a suite of 5 different extreme weather environments parameterized by 5 different ϕ weather configurations, as well as ϕ_0 for a total of 6 environments. The agent is trained on automatically generated environments according to each UED algorithm. It is then evaluated in the following 6 environments: ϕ_0 simulates realistic weather based on recordings from Arizona, USA, ϕ_1 simulates an extremely hot and dry drought, ϕ_2 simulates a wet and windy storm, ϕ_3 simulates a humid heatwave, ϕ_4 simulates a cold snap, and ϕ_5 simulates erratic weather. **Our hypothesis is that using uncertainty to identify new environments to collect data from will allow us to train RL agents that are more robust to extreme weather conditions.**

In order to test whether or not the RL policies trained in simulation can be extended to the real world, we evaluated each RL algorithm in each of the 6 environments by running the EnergyPlus simulator at a higher fidelity than the agents were trained on, thus simulating the "Sim2Real" jump with a more realistic simulator. The agents were trained on a simulator operating at a granularity of $\Delta t = 1$hour per timestep, and we evaluate on a granularity of $\Delta t = 0.25$hours per timestep. During evaluation, each of the RL agents' actions are simply repeated four times so that it still takes an action every hour. **Our hypothesis is that by increasing the state space supported by the training distribution, ActivePLR will help the agent be robust to compounding errors caused by the Sim2Real jump.**

Results and Discussion

ActivePLR is Robust to Extreme Weather Events

In order to evaluate how robust agents trained by each algorithm are to extreme weather, we evaluate the agent in a suite of 6 different environments, parameterized by 5 different ϕ extreme weather configurations and ϕ_0. Figure 4 shows the overall performance of each algorithm. See the appendix[12] for performance in specific environments.

Surprisingly, DR and RPLR did not have significant improvements over the vanilla RL algorithm. By the end of training, DR and RPLR achieved 9% higher reward than the RBC on the base environment ϕ_0 after 3M timesteps of training. However, the vanilla RL policy had a 8% improvement over the DR and RPLR policies with ϕ_0. Over the 5 extreme weather environments, DR did about as well as the RBC. The unexpected lack of performance gain may mean the environments generated with DR were too unrealistic to learn how to perform in extreme weather conditions.

Over the extreme weather conditions, RPLR beat DR and the RBC, but performed worse than the vanilla RL policy. This indicates that as RPLR uses DR to sample new environments, it may still suffer from generating unrealistic environments. However, its weighted environment resampling procedure helps it generalize better than naive DR, even though it is resampling from unrealistic environments. We found that training the HVAC controller with ActivePLR resulted in agents that performed better in both the extreme environments and the base environment.

Generally, ActiveRL and ActivePLR performed similarly. In three out of the five extreme environments: (1) the hot drought, (2) the cold and windy, and (3) the cold snap environment, ActivePLR performed significantly better than all other baselines and performed competitively in the remaining two environments. In the base environment, ActivePLR provides a 9% improvement over vanilla RL, and a 24% improvement over RBC. Over all 6 environments, it provides a 3% improvement over vanilla RL. We also found that over the 6 environments, ActivePLR and ActiveRL provided a 15% relative decrease in days with ASHRAE thermal comfort violations compared to vanilla RL (which was the best baseline in terms of thermal comfort), resulting in significantly more comfortable occupants even during extreme weather conditions.[15] The fact that ActivePLR significantly outperforms all baselines indicates there is considerable value in seeking out realistic new training environments that maximize an agent's uncertainty rather than choosing environments at random or merely replaying old ones.

ActivePLR Generalizes from Simulation

One flaw in this work and UED algorithms in general is that a simulator is required to train the model in different environments. Thus, it is important to ask whether or not the RL policies trained in simulation can be extended to the real world. In order to approximate the Sim2Real gap, we conducted evaluated each RL algorithm in each of the 6 environments by running the EnergyPlus simulator at a higher fidelity than the agents were trained on, so that these test environments (1) had slightly different dynamics from the original training simulation, which should give rise to similar distribution shift issues as the Sim2Real gap, and (2) had dynamics that were as close to those of the real world as possible because the test simulation was run with higher fidelity and should therefore be more accurate to real dynamics than the training simulation. An illustration of the performances of each algorithm on each of the 6 handcrafted environments are shown in Figure 4.

When the agents are transferred from the simulation to our surrogate for the real world, we see there is a significant performance drop across all data-driven algorithms. Vanilla RL achieves a reward that is 8.5% lower on average across the 6 handcrafted environments when evaluated on the higher fidelity simulation. DR and RPLR have smaller relative drops of about 7%. However, since the 7% is relative to the performance of DR and RPLR in the original low fidelity sim-

---

[15] ASHRAE defines uncomfortable thermal conditions as at least 20% of occupants are predicted to be uncomfortable (PPD $> 20\%$)

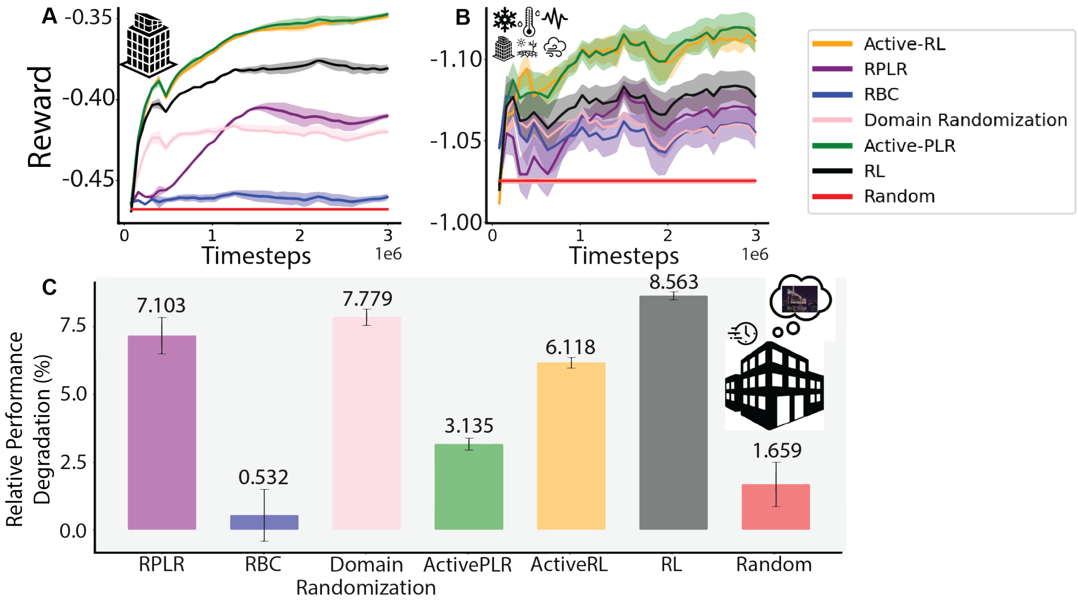

Figure 4: Performance of each algorithm on training an RL HVAC agent in Sinergym, tested on various weather patterns. ActiveRL and ActivePLR outperform all baselines. We report the standard error of the mean over 5 trials for each result. Each ActiveRL and ActivePLR trial took 12 hours to train and evaluate. A. Average reward of each algorithm on the base environment ϕ_0 throughout training. B. Average reward achieved by each algorithm averaged over all 6 environments throughout training. C. The average drop in reward over all environments when evaluating in the higher fidelity simulation compared to evaluation in the lower fidelity simulation. Lower is better here. Note that none of the algorithms were trained on the specific extreme weather environments. Evaluating ActiveRL and ActivePLR on the higher fidelity simulation took about 1 hour per trial.

ulation, vanilla RL still performs better in terms of absolute reward over the six environments. Random and RBC have very small or no relative performance degradation, which is to be expected because they are not data-driven models. They still perform the worst in terms of absolute reward. ActiveRL and ActivePLR, however, achieve both smaller relative drops in performance and higher absolute reward across all the different extreme weather scenarios. ActiveRL has only a 6.1% relative drop in reward while ActivePLR has only a 3.1% relative drop. These are promising results that indicate that these algorithms would still perform well if deployed in the real world after being trained in simulation. Furthermore, these algorithms result in agents that are more robust to the Sim2Real transfer than other methods.

We found that ActivePLR trains agents that are more robust to the Sim2Real transfer than ActiveRL, which is surprising since ActivePLR was not significantly different from ActiveRL in the extreme weather experiment. There might be some attractive local optimum in the HVAC control task in the low fidelity simulation that both ActivePLR and ActiveRL fall into that is not present in the high fidelity simulation, resulting in similar performance in the extreme weather experiments but better Sim2Real transfer. In addition, the recorded value loss that RPLR and ActivePLR use is likely a less noisy signal of environment curriculum value than the uncertainty over the value estimate that ActiveRL uses. The value loss is obtained by actually collecting data while the value uncertainty is estimated using only the model weights through Monte Carlo Dropout.

Conclusion and Limitations

We explored the utility of a novel uncertainty-driven, gradient based algorithm called ActivePLR for unsupervised environment design in the context of training RL building control agents that are robust to climate change. We found that incorporating uncertainty into UED through ActivePLR led to HVAC controllers that better optimized thermal comfort and energy usage, even in extreme weather scenarios that were never in the training distribution. Our experiments showed that other UED algorithms perform poorly when generating new environment configurations for weather patterns because they may output unrealistic weather patterns that do not help the RL agent perform well in more realistic weather scenarios. Furthermore, we showed that ActivePLR and its variant ActiveRL would have a much smaller degradation in performance when transferring from the simulated domain to the real world compared to other techniques, making them a practical option for training robust RL HVAC agents that are ready for real deployment.

This work has two primary limitations. The first is that we rely on simulations; this is a flaw that is shared by most work that focuses on UED as well as many works in building control, as access to real buildings is difficult to obtain. The second is that our method requires continuous environment configuration variables to conduct gradient ascent. Future work could explore how this could be mitigated by applying dequantization techniques (Das and Spanos 2022) that transform categorical variables into continuous variables.

Acknowledgements

This research is funded by the Republic of Singapore's National Research Foundation through a grant to the Berkeley Education Alliance for Research in Singapore (BEARS) for the Singapore-Berkeley Building Efficiency and Sustainability in the Tropics (SinBerBEST) Program. BEARS has been established by the University of California, Berkeley as a center for intellectual excellence in research and education in Singapore.

References

An, G.; Moon, S.; Kim, J.-H.; and Song, H. O. 2021. Uncertainty-based offline reinforcement learning with diversified q-ensemble. *Advances in neural information processing systems*, 34: 7436–7447.

ANSI, A.; and ASHRAE, M. 2017. Standard 55—thermal environmental conditions for human occupancy. *Amer. Soc. Heat, Refrigerat. Air Condition. Eng*, 1451992.

Bouneffouf, D. 2016. Exponentiated gradient exploration for active learning. *Computers*, 5(1): 1.

Brockman, G.; Cheung, V.; Pettersson, L.; Schneider, J.; Schulman, J.; Tang, J.; and Zaremba, W. 2016. Openai gym. arXiv. *arXiv preprint arXiv:1606.01540*, 10.

Chen, B.; Cai, Z.; and Bergés, M. 2019. 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*, 316–325.

Chen, X.; Hu, J.; Jin, C.; Li, L.; and Wang, L. 2021. Understanding domain randomization for sim-to-real transfer. *arXiv preprint arXiv:2110.03239*.

Cohn, D.; Atlas, L.; and Ladner, R. 1994. Improving generalization with active learning. *Machine learning*, 15: 201–221.

Crawley, D. B.; Lawrie, L. K.; Winkelmann, F. C.; Buhl, W. F.; Huang, Y. J.; Pedersen, C. O.; Strand, R. K.; Liesen, R. J.; Fisher, D. E.; Witte, M. J.; et al. 2001. EnergyPlus: creating a new-generation building energy simulation program. *Energy and buildings*, 33(4): 319–331.

Das, H. P.; Lin, Y.-W.; Agwan, U.; Spangher, L.; Devonport, A.; Yang, Y.; Drgona, J.; Chong, A.; Schiavon, S.; and Spanos, C. J. 2022. Machine Learning for Smart and Energy-Efficient Buildings. *arXiv preprint arXiv:2211.14889*.

Das, H. P.; and Spanos, C. J. 2022. Improved dequantization and normalization methods for tabular data pre-processing in smart buildings. In *Proceedings of the 9th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation*, 168–177.

Deng, X.; Zhang, Y.; and Qi, H. 2022. Towards optimal HVAC control in non-stationary building environments combining active change detection and deep reinforcement learning. *Building and environment*, 211: 108680.

Dennis, M.; Jaques, N.; Vinitsky, E.; Bayen, A.; Russell, S.; Critch, A.; and Levine, S. 2020. Emergent complexity and zero-shot transfer via unsupervised environment design. *Advances in neural information processing systems*, 33: 13049–13061.

Doob, J. L. 1942. The Brownian movement and stochastic equations. *Annals of Mathematics*, 351–369.

Fanger, P. O. 1967. Calculation of thermal comfort: introduction of a basic comfort equation. *ASHRAE Trans, Part II*, 73: III4–1.

Faria, B.; Perdigão, D.; Brás, J.; and Macedo, L. 2022. The Joint Role of Batch Size and Query Strategy in Active Learning-Based Prediction-A Case Study in the Heart Attack Domain. In *Progress in Artificial Intelligence: 21st EPIA Conference on Artificial Intelligence, EPIA 2022, Lisbon, Portugal, August 31–September 2, 2022, Proceedings*, 464–475. Springer.

Gal, Y.; and Ghahramani, Z. 2016. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, 1050–1059. PMLR.

Gallego-Posada, J.; and Ramirez, J. 2022. Cooper: a toolkit for Lagrangian-based constrained optimization. https://github.com/cooper-org/cooper. Accessed: 2024-02-06.

Gong, K.; Yang, J.; Wang, X.; Jiang, C.; Xiong, Z.; Zhang, M.; Guo, M.; Lv, R.; Wang, S.; and Zhang, S. 2022. Comprehensive review of modeling, structure, and integration techniques of smart buildings in the cyber-physical-social system. *Frontiers in Energy*, 1–21.

Gunn, S.; Jang, D.; Paradise, O.; Spangher, L.; and Spanos, C. J. 2022. Adversarial poisoning attacks on reinforcement learning-driven energy pricing. In *Proceedings of the 9th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation*, 262–265.

Hayat, H.; Griffiths, T.; Brennan, D.; Lewis, R. P.; Barclay, M.; Weirman, C.; Philip, B.; and Searle, J. R. 2019. The state-of-the-art of sensors and environmental monitoring technologies in buildings. *Sensors*, 19(17): 3648.

Jang, D.; Spangher, L.; Srivistava, T.; Khattar, M.; Agwan, U.; Nadarajah, S.; and Spanos, C. 2021. Offline-online reinforcement learning for energy pricing in office demand response: lowering energy and data costs. In *Proceedings of the 8th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation*, 131–139.

Jiang, M.; Dennis, M.; Parker-Holder, J.; Foerster, J.; Grefenstette, E.; and Rocktäschel, T. 2021. Replay-guided adversarial environment design. *Advances in Neural Information Processing Systems*, 34: 1884–1897.

Jiang, M.; Dennis, M.; Parker-Holder, J.; Lupu, A.; Küttler, H.; Grefenstette, E.; Rocktäschel, T.; and Foerster, J. 2022. Grounding Aleatoric Uncertainty in Unsupervised Environment Design. *arXiv preprint arXiv:2207.05219*.

Jiang, M.; Grefenstette, E.; and Rocktäschel, T. 2021. Prioritized level replay. In *International Conference on Machine Learning*, 4940–4950. PMLR.

Jiménez-Raboso, J.; Campoy-Nieves, A.; Manjavacas-Lucas, A.; Gómez-Romero, J.; and Molina-Solana, M. 2021. Sinergym: A Building Simulation and Control Framework for Training Reinforcement Learning Agents. In *Proceedings of the 8th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation*, 319–323. New York, NY, USA: Association for Computing Machinery. ISBN 9781450391146.

Kingma, D. P.; and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Korpelevich, G. M. 1976. The extragradient method for finding saddle points and other problems. *Matecon*, 12: 747–756.

Kou, X.; Du, Y.; Li, F. F.; Pulgar-Painemal, H.; Zandi, H.; Dong, J.; and Olama, M. M. 2021. Model-Based and Data-Driven HVAC Control Strategies for Residential Demand Response. *IEEE Open Access Journal of Power and Energy*, 8: 186–197.

Kurte, K. R.; Munk, J. D.; Kotevska, O.; Amasyali, K.; Smith, R.; Mckee, E.; Du, Y.; Cui, B.; Kuruganti, T.; and Zandi, H. 2020. Evaluating the Adaptability of Reinforcement Learning Based HVAC Control for Residential Houses. *Sustainability*.

Lakshminarayanan, B.; Pritzel, A.; and Blundell, C. 2017. Simple and scalable predictive uncertainty estimation using deep ensembles. *Advances in neural information processing systems*, 30.

Lissa, P.; Schukat, M.; and Barrett, E. 2020. Transfer learning applied to reinforcement learning-based hvac control. *SN Computer Science*, 1: 1–12.

Lynch, D. P. 2003. EVOP design of experiments. Technical report, Citeseer.

Makili, L. E.; Sánchez, J. A. V.; and Dormido-Canto, S. 2012. Active learning using conformal predictors: application to image classification. *Fusion Science and Technology*, 62(2): 347–355.

Masson-Delmotte, V.; Zhai, P.; Pirani, A.; Connors, S. L.; Péan, C.; Berger, S.; Caud, N.; Chen, Y.; Goldfarb, L.; Gomis, M.; et al. 2021. Climate change 2021: the physical science basis. *Contribution of working group I to the sixth assessment report of the intergovernmental panel on climate change*, 2.

Mathews, E. H.; Botha, C. P.; Arndt, D. C.; and Malan, A. G. 2001. HVAC control strategies to enhance comfort and minimise energy usage. *Energy and Buildings*, 33: 853–863.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; and Riedmiller, M. 2013. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

Naug, A.; Quinones-Grueiro, M.; and Biswas, G. 2022. Deep reinforcement learning control for non-stationary building energy management. *Energy and Buildings*, 277: 112584.

Parker-Holder, J.; Jiang, M.; Dennis, M.; Samvelyan, M.; Foerster, J.; Grefenstette, E.; and Rocktäschel, T. 2022. Evolving curricula with regret-based environment design. In *International Conference on Machine Learning*, 17473–17498. PMLR.

Risi, S.; and Togelius, J. 2020. Increasing generality in machine learning through procedural content generation. *Nature Machine Intelligence*, 2(8): 428–436.

Rizvi, S. A. A.; and Pertzborn, A. J. 2022. Experimental Results of a Disturbance Compensating Q-learning Controller for HVAC Systems. In *2022 American Control Conference (ACC)*, 3353–3353. IEEE.

Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Silver, D.; Schrittwieser, J.; Simonyan, K.; Antonoglou, I.; Huang, A.; Guez, A.; Hubert, T.; Baker, L.; Lai, M.; Bolton, A.; et al. 2017. Mastering the game of go without human knowledge. *nature*, 550(7676): 354–359.

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

Tobin, J.; Fong, R.; Ray, A.; Schneider, J.; Zaremba, W.; and Abbeel, P. 2017. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, 23–30. IEEE.

U.S. Department of Energy—EIA. 2020. Annual Energy Outlook 2020. Table 18. Energy-Related Carbon Dioxide Emissions by Sector and Source. https://www.eia.gov/outlooks/aeo.

Vuong, Q.; Vikram, S.; Su, H.; Gao, S.; and Christensen, H. I. 2019. How to pick the domain randomization parameters for sim-to-real transfer of reinforcement learning policies? *arXiv preprint arXiv:1903.11774*.

Wang, P.; Bouaynaya, N. C.; Mihaylova, L.; Wang, J.; Zhang, Q.; and He, R. 2020. Bayesian neural networks uncertainty quantification with cubature rules. In *2020 International Joint Conference on Neural Networks (IJCNN)*, 1–7. IEEE.

Wemhoff, A.; and Frank, M. 2010. Predictions of energy savings in HVAC systems by lumped models. *Energy and Buildings*, 42(10): 1807–1814.

Wu, Y.; Zhai, S.; Srivastava, N.; Susskind, J.; Zhang, J.; Salakhutdinov, R.; and Goh, H. 2021. Uncertainty weighted actor-critic for offline reinforcement learning. *arXiv preprint arXiv:2105.08140*.

Xu, S.; Wang, Y.; Wang, Y.; O'Neill, Z.; and Zhu, Q. 2020. One for many: Transfer learning for building hvac control. In *Proceedings of the 7th ACM international conference on systems for energy-efficient buildings, cities, and transportation*, 230–239.

Zhang, C.; Vinyals, O.; Munos, R.; and Bengio, S. 2018. A study on overfitting in deep reinforcement learning. *arXiv preprint arXiv:1804.06893*.

Zhang, X.; Chintala, R.; Bernstein, A.; Graf, P.; and Jin, X. 2021. Grid-interactive multi-zone building control using reinforcement learning with global-local policy search. In *2021 American Control Conference (ACC)*, 4155–4162. IEEE.