SustainGym: Reinforcement Learning Environments for Sustainable Energy Systems

Christopher Yeh[1], **Victor Li**[1], **Rajeev Datta**[1], **Julio Arroyo**[1], **Nicolas Christianson**[1], **Chi Zhang**[2], **Yize Chen**[3], **Mehdi Hosseini**[4], **Azarang Golmohammadi**[4], **Yuanyuan Shi**[2], **Yisong Yue**[1], **Adam Wierman**[1]
[1]California Institute of Technology
[2]University of California, San Diego
[3]Hong Kong University of Science and Technology
[4]Beyond Limits
[1]{cyeh,vhli,rdatta,jarroyoi,yyue,adamw}@caltech.edu
[2]chz056@ucsd.edu, yyshi@eng.ucsd.edu
[3]yizechen@ust.hk
[4]{mhosseini, agolmohammadi}@beyond.ai

Abstract

The lack of standardized benchmarks for reinforcement learning (RL) in sustainability applications has made it difficult to both track progress on specific domains and identify bottlenecks for researchers to focus their efforts. In this paper, we present SustainGym, a suite of five environments designed to test the performance of RL algorithms on realistic sustainable energy system tasks, ranging from electric vehicle charging to carbon-aware data center job scheduling. The environments test RL algorithms under realistic distribution shifts as well as in multi-agent settings. We show that standard off-the-shelf RL algorithms leave significant room for improving performance and highlight the challenges ahead for introducing RL to real-world sustainability tasks.

1 Introduction

While reinforcement learning (RL) algorithms have demonstrated tremendous success in applications ranging from game-playing, *e.g.*, Atari and Go, to robotic control, *e.g.*, [1–3], most RL algorithms continue to only be benchmarked using toy environments—*e.g.*, OpenAI Gym [4]. These toy environments generally do not have realistic physical constraints, nor realistic environmental shifts over time. Furthermore, these environments are generally limited to single-agent systems, whereas real-world systems tend to involve coordination and/or competition between actors. The realism gap limits the reliable deployment of off-the-shelf RL algorithms in real-world systems.

Developing better RL algorithms to address these challenges requires a means of empirically benchmarking and comparing the performance of different algorithms in real-world settings. Our inspiration comes from progress in supervised machine learning (ML), where widespread adoption of breakthrough techniques was fueled by large datasets with standardized benchmarks, such as ImageNet for computer vision [5] and the GLUE benchmark for natural language processing [6]. More recently, many supervised learning datasets have been created to address specific real-world sustainability challenges, such as monitoring global progress towards sustainable development goals [7].

In this work, we introduce SustainGym, a suite of 5 RL environments that realistically model sustainability settings, summarized in Table 1:

37th Conference on Neural Information Processing Systems (NeurIPS 2023) Track on Datasets and Benchmarks.

Table 1: Summary of environments included in SustainGym and their features. The "Single agent" and "Multi-agent" rows indicate what an individual RL agent controls in that environment.

Env	EVChargingEnv	ElectricityMarketEnv	DatacenterEnv	CogenEnv	BuildingEnv
Control task	charging rates for EV charging stations	market bids for a grid-connected battery storage system	virtual capacity curve for a carbon-aware data center	dispatch set points for turbines	heating supply for buildings
Modeled after	charging networks at Caltech & JPL	generic test case (IEEE RTS-GMLC)	(loosely) a Google data center	specific combined cycle gas generation plant in the U.S.	generic DoE commercial reference building models
Single agent	all EV charging stations	single battery system	single data center	all 4 turbine units	all buildings
Multi-agent	one EV charging station, cooperative	N/A	N/A	one turbine unit, cooperative	one room, cooperative
Actions	discrete or continuous	discrete or continuous	continuous	mixed discrete & continuous	discrete or continuous
Rewards	cost + CO_2	cost + CO_2	penalty + CO_2	cost + CO_2	temperature difference + energy use
Distribution shift	MOER, EV arrivals	MOER, load	MOER	renewable wind penetration	outdoor temperature

- `EVChargingEnv` models the problem of scheduling electric vehicle (EV) charging to meet user needs while minimizing CO_2 emissions.
- `ElectricityMarketEnv` models a grid-scale battery storage system bidding into the electricity market to generate profit (through price arbitrage) and reduce CO_2 emissions.
- `DatacenterEnv` models a datacenter deciding on a "virtual capacity curve" to shift flexible jobs towards times of day with lower CO_2 emissions.
- `CogenEnv` models a combined cycle cogeneration plant producing steam and electricity to meet local demand while minimizing fuel usage and ramp costs.
- `BuildingEnv` models the thermal control of building energy systems to reduce the total electricity consumption while satisfying the user-specified temperature requirement.

A key feature of SustainGym environments is their support for testing RL algorithms under realistic and natural exogenous distribution shifts, which generally fall under two categories:

1. *Shifts in demand*. In each environment, RL agents choose actions to satisfy some "demand" that is often affected by the behavior of unmodeled agents. For example, in `EVChargingEnv`, the demand is the amount of energy that needs to be delivered to EVs that have arrived at the charging network. This demand changed significantly at the start of the COVID-19 pandemic when EV drivers changed their driving behaviors (Figure 2).
2. *Shifts in environmental parameters*. Real-world environments are rarely static, and SustainGym environments reflect changing environment parameters due to temporal and/or climate changes. For example, a battery storage controller for `ElectricityMarketEnv` makes decisions to minimize marginal CO_2 emissions, but the distribution of CO_2 emissions varies over time as power plants are added to or removed from the electric grid (Figure 1).

Notably, the distribution shifts reflected in SustainGym are unlike the "sim-to-real" or offline-vs-online RL distribution shifts that have been more commonly studied in the literature. The sim-to-real distribution shift comes from imperfect modeling of the environment, whereas the offline-to-online RL distribution shift is caused by a change in the policy used to generate trajectories. In contrast, the exogenous distribution shifts in SustainGym are not due to imperfect environments nor policy mismatches, but rather more fundamental changes in the transition dynamics of the Markov decision processes. Note that only the transition dynamics experience distribution shift; the state space, action space, and reward functions do not change.

Two other similar lines of work to the distribution shifts in SustainGym are nonstationary RL environments [8] and distributionally robust RL [9]. However, whereas nonstationary RL typically

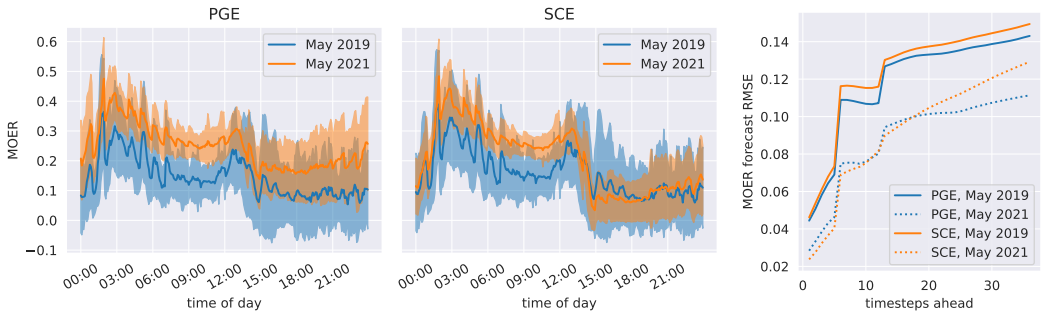


Figure 1: (left) MOER values from two different regions (a.k.a. "balancing authorities") in California, Pacific Gas & Electric (PGE) and Southern California Edison (SCE). Solid line is the mean MOER over all days in a month at a given time of day. Shaded region is ± 1 std. dev. (right) MOER forecast error increases with the forecast horizon.

evaluates an RL agent's performance over the course of a changing environment, SustainGym benchmarks RL agents' ability to generalize to new (unseen) distribution shifts. Distributionally robust RL generally assumes that the set of environmental distributions are known at training time, which is not necessarily the case for the settings considered in SustainGym.

In addition to modeling realistic distribution shifts, SustainGym is distinctive for its inclusion multi-agent interactions, physical constraints, and mixtures of discrete and continuous actions, as summarized in Table 1.

To demonstrate the use of the SustainGym, we perform experiments with off-the-shelf RL algorithms. We find that these algorithms have mixed performance on SustainGym. Furthermore, we show that distribution shifts may reduce the performance of these algorithms significantly, demonstrating a need for more robust algorithms. Finally, comparisons against non-RL baselines and oracles show that RL has significant room for improvement.

Due to page constraints, the main text of this paper summarizes key design choices and experimental observations for SustainGym. Details can be found in Appendix B. Code, licenses, and instructions for using SustainGym can be found on GitHub.[1]

Related Work. Prior work related to SustainGym includes ConservationGym, which focuses on ecological applications [10], PowerGridWorld for power system modeling and simulation [11], and CityLearn for simulation of demand response and urban energy management [12], among others. RL environments and algorithms for both EV charging [13–15] and electricity markets [16–18] have also been released. Compared to these works, the unique aspects of SustainGym are its focus on tracking estimated CO_2 emissions and its ability to test RL algorithms in settings with challenging distribution shifts, physical constraints, and interactions between multiple agents. We expect SustainGym to serve as a benchmark for the progress of RL algorithm development for sustainable energy systems.

2 Environments

This section introduces the 5 environments in SustainGym and summarizes their design choices.

Marginal CO_2 emissions. Three environments (`EVChargingEnv`, `ElectricityMarketEnv`, `DatacenterEnv`) impose a cost P_{CO_2} (in \$/kgCO_2) on the simulated CO_2 emissions induced by the actions of an agent as a result of changes in electricity consumption. To do so, our environments use data on California's historical marginal operating emissions rate (MOER, in kgCO_2/kWh), which is the increase in CO_2 emissions per increase in energy demand. The MOER at time t is denoted $m_t \in \mathbb{R}_+$, and the forecasts generated at time t for the next k time steps are denoted $\hat{m}_{t:t+k-1|t} \in \mathbb{R}^k$. By default, we use $k = 36$. Figure 1 shows how MOER values and their forecasts vary across time and between different regions in California.

[1] `https://github.com/chrisyeh96/sustaingym/`

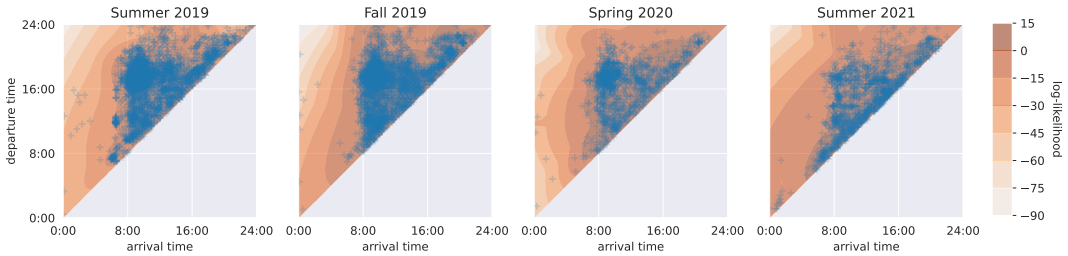


Figure 2: EV arrival vs. departure times for the Caltech EV charging network. Historical data is in blue, and log-likelihood contours from a 30-component GMM are in orange. The distribution of EV arrival and departure times changed noticeably when the COVID-19 pandemic began in early 2020.

2.1 `EVChargingEnv`

`EVChargingEnv` uses ACNSim [13] to simulate the charging of EVs based on actual data gathered from EV charging networks between fall 2018 and summer 2021 [19, 20]. ACNSim is a "digital twin" of actual EV charging networks at Caltech and JPL, which have $n = 54$ and 52 charging stations (abbrv. EVSEs, Electric Vehicle Supply Equipment), respectively. ACNSim accounts for nonlinear EV battery charging dynamics and unbalanced 3-phase AC power flows, and is thus very realistic. ACNSim (and therefore `EVChargingEnv`) can be extended to model other charging networks as well. When drivers charge their EVs, they provide an estimated time of departure and amount of energy requested. Because of network and power constraints, not all EVSEs can simultaneously provide their maximum charging rates (a.k.a. "pilot signals").

Each episode starts at midnight and runs at 5-minute time steps for 24 hours. At each time step, the agent simultaneously decides all n EVSE pilot signals to be executed for the duration of that time step. Its objective is to maximize charge delivery while minimizing carbon costs and obeying the network and power constraints.

Observation Space. An observation at time t is $s(t) = (t, d, e, m_{t-1}, \hat{m}_{t:t+k-1|t})$. $t \in [0, 1]$ is the fraction of day. $d \in \mathbb{Z}^n$ is estimated remaining duration of each EV (in # of time steps). $e \in \mathbb{R}^n_+$ is remaining energy demand of each EV (in kWh). If no EV is charging at EVSE i, then $d_i = 0$ and $e_i = 0$. If an EV charging at EVSE i has exceeded the user-specified estimated departure time, then d_i becomes negative, while e_i may still be nonzero.

Action Space. `EVChargingEnv` exposes a choice of discrete actions $a(t) \in \{0, 1, 2, 3, 4\}^n$, representing pilot signals scaled down by a factor of 8, or continuous actions $a(t) \in [0, 1]^n$ representing the pilot signal normalized by the maximum signal allowed M (in amps) for each EVSE. Physical infrastructure in a charging network constrains the set $\mathcal{A}_t$ of feasible actions at each time step t [20]. Furthermore, the EVSEs only support discrete pilot signals, so $\mathcal{A}_t$ is nonconvex. To satisfy these physical constraints, `EVChargingEnv` can project an agent's action $a(t)$ into the convex hull of $\mathcal{A}_t$ and round it to the nearest allowed pilot signal, resulting in final normalized pilot signals $\tilde{a}(t)$. ACNSim processes $\tilde{a}(t)$ and returns the actual charging rate $M\tilde{a} \in \mathbb{R}^n_+$ (in amps) delivered at each EVSE, as well as the remaining demand $e_i(t + 1)$.

Reward Function. The reward function is a sum of three components: $r(t) = p(t) - c_V(t) - c_C(t)$. The profit term $p(t)$ aims to maximize energy delivered to the EVs. The constraint violation cost $c_V(t)$ penalizes network and power constraint violations. Finally, the CO_2 emissions cost $c_C(t)$, which is a function of the MOER m_t and charging action, aims to reduce emissions by encouraging the agent to charge EVs when the MOER is low.

Distribution Shift. `EVChargingEnv` supports real historical data as well as data sampled from a 30-component Gaussian Mixture Model (GMM) fit to historical data. We fitted GMMs to 4 disjoint historical periods, as defined in [21]. Figures 2 and 6 show the distribution of arrival and departure times in each of these 4 periods, for both the historical data as well as the GMM log-likelihoods. From these figures, it is evident that the pattern of user arrival and departure times changes over time, with the most drastic shift happening between Fall 2019 and Spring 2020, which is when the COVID-19 pandemic began.

Multiagent Setting. The multiagent setting features n agents, each deciding the pilot signal for a single EVSE. The reward is split evenly among the agents. Each agent obtains the global observation, except that the estimated remaining durations and energy demands for other EVSEs are delayed by t_d time steps.

2.2 `ElectricityMarketEnv`

`ElectricityMarketEnv` simulates a realtime electricity market for 33 generators and 1 80MWh battery storage system connected on a 24-bus congested transmission network based on the widely-used IEEE RTS-24 test case [22], with 5-minute settlements and load data from IEEE RTS-GMLC [23]. While `ElectricityMarketEnv` is not modeled after any particular real-world transmission network, the RTS-GMLC electricity load profile was designed to be representative of a modern transmission network located in the southwestern U.S.

All participants submit bids to the market operator (MO) at every time step. Based on the bids, the MO solves the multi-timestep security-constrained economic dispatch (SCED) problem which determines the price and amount of electricity purchased from (or sold by) each generator and battery to meet realtime electricity demand. Each episode runs for 1 day, with 5-minute time intervals. The agent controls the battery system and is rewarded for submitting bids that result in charging (buy) when prices are low, and discharging (sell) when prices and CO_2 emissions are high, thus performing price arbitrage.

Observation Space. An observation is $s(t) = (t, e, a(t-1), x_{t-1}, p_{t-1}, l_{t-1}, \hat{l}_{t:t+k-1}, m_{t-1}, \hat{m}_{t:t+k-1|t})$. $t \in [0, 1]$ represents the time of day. $e \in \mathbb{R}_+$ is the agent's battery level (in MWh). $a(t-1) \in \mathbb{R}_+^{2\times k}$ is the previous action taken. $x_{t-1} \in \mathbb{R}$ is the previous dispatch (in MWh) asked of the agent. $p_{t-1} \in \mathbb{R}_+$ is the previous price experienced by the agent (in \$/MWh). $l_{t-1} \in \mathbb{R}_+$ is the previous demand experienced by the agent (in MWh), while $\hat{l}_{t:t+k-1} \in \mathbb{R}^k$ is the forecasted demand for the next k steps.

Action Space. An agent action is a bid $a(t) = (a^c, a^d) \in \mathbb{R}_+^k \times \mathbb{R}_+^k$, representing prices (\$/MWh) that the agent is willing to pay (or receive) for charging (or discharging) per MWh of energy, for the next $k+1$ time steps starting from time t. The generators are assumed to always bid their fixed true cost of generation. The environment solves the optimal dispatch problem to determine the electricity price p_t and the agent's dispatch $x_t \in \mathbb{R}$, which is the amount of energy that the agent is obligated to sell into or buy from the grid within the next time step. The dispatch in turn determines the storage system's next energy level. We also provide a wrapper that discretizes the action space into 3 actions only: charge, do nothing, or discharge.

Reward Function. The reward function encourages the agent to maximize profit from charging decisions while minimizing associated carbon emissions. It is a sum of three components: $r(t) = r_R(t) + r_C(t) - c_T(t)$. The revenue term $r_R(t) = p_t x_t$ is the immediate revenue from the dispatch. The CO_2 emissions reward term $r_C(t) = P_{CO_2} m_t x_t$ represents the price of CO_2 emissions displaced or incurred by the battery dispatch. The terminal cost $c_T(t)$, which is nonzero only when $t = T$, encourages the battery to have the same energy level at the end of the day as when it started. We also provide an option to delay all reward signals until the terminal time (intermediate rewards are 0).

Distribution Shift. Distribution shift for `ElectricityMarketEnv` comes from changes in both electricity demand and MOER profiles between summer and winter months.

2.3 `DatacenterEnv`

`DatacenterEnv` is a simulator for carbon-aware job scheduling in datacenters, which aims to reduce the carbon emissions associated with electrcity usage in a datacenter. Carbon-aware job scheduling is premised upon two facts: (i) a significant fraction of a datacenter's workload (*e.g.*, up to 50% in some of Google datacenters [24, 25]) is comprised of low priority jobs whose execution can be delayed, and (ii) the carbon intensity of the electric grid fluctuates predictably over time. Therefore, if the execution of low priority workload is delayed to a time of day with "greener" energy, the datacenter's carbon emissions can be minimized.

`DatacenterEnv` is loosely modeled after a Google datacenter. We assume that jobs are scheduled according to a priority queue, with jobs spread evenly across the available machines. Following

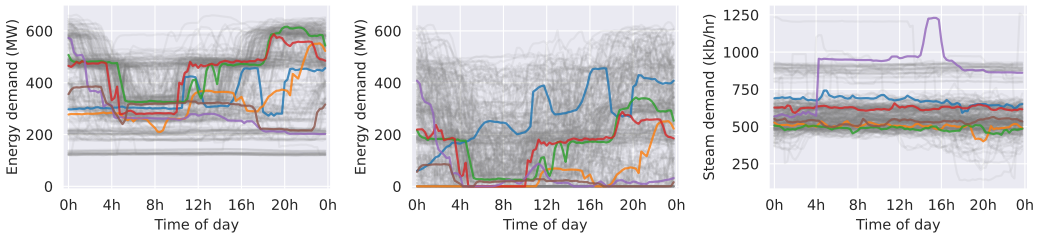


Figure 3: Electricity demand without (left) and with wind (middle), and steam demand (right) on a 15 minute basis for 253 days in `CogenEnv` dataset, with 6 random traces highlighted for each.

Radovanovic et al. [26], we implement workload execution delay by artificially limiting the total datacenter capacity with a *virtual capacity curve* (VCC) at each time step. If more jobs are enqueued than the VCC permits, then the jobs must wait until the VCC is raised high enough to allow the jobs to run. Simulation is carried out by replaying a sample of real job traces from a Google cluster from May 2019 [25]. One timestep in the environment corresponds to one hour, and each episode lasts the whole month.

Observation Space. An observation $s(t) = (a(t-1), d_t, n, \hat{m}_{t:t+23|t}) \in \mathbb{R}^{27}$ contains the active VCC $a(t-1)$ set from the previous time step, currently running compute load d_t, number of jobs waiting to be scheduled n, as well as the forecasted MOER for the next 24h $\hat{m}_{t:t+23|t}$.

Action Space. At time t, the agent sets the VCC, $a(t) \in [0, 1]$, for the next time step. This action denotes the *fraction* of the datacenter's maximum capacity allowed to be allocated by the scheduler.

Reward Function. The reward consists of two components that encourage the agents to trade-off between scheduling more jobs and reducing associated carbon emissions. The first component penalizes the agent when jobs are scheduled more than 24h after they were originally submitted. The second component is a carbon emissions cost. Formally, the reward is specified as

$$r(t) = d_t \cdot m_t + \mathbf{1}_{[t\%24=0]} \max\left(0, 0.97 w_t - C \sum_{h=0}^{23} a(t-h)\right)$$

where $d_t \cdot m_t$ is the carbon emissions, C is the datacenter's maximum capacity, and w_t is the total job-hours of enqueued jobs on that day.

Distribution Shift. The distribution shift in `DatacenterEnv` comes from changes in the MOER between 2019 and 2021.

2.4 `CogenEnv`

`CogenEnv` simulates the operation of a combined cycle gas power plant tasked with meeting local steam and energy demand. Conventional dispatchable generators suffer decreased efficiency as a result of frequent ramping, posing a particular challenge as increasing penetrations of variable renewables necessitate larger and more frequent ramps to ensure supply-demand balance. Thus, optimal operation of cogeneration resources requires balancing the competing objectives of minimizing fuel use, anticipating future ramp needs, and ensuring delivery of sufficient energy and steam to the grid.

While `CogenEnv` models a specific combined cycle gas generation plant in the U.S. (anonymized and location withheld for security reasons), the basic environment setup is a representative prototype of more general dispatchable resource generation control tasks, due to its complexity (the number of variables, mixed continuous/binary decisions, complementary trains of the plant all needing to be controlled together). In addition, the environment is readily modifiable to accommodate other cost structures (*e.g.*, changing the relative magnitude of the constraint penalties vs. the ramping cost).

Observation Space. An observation takes the form

$$s(t) = (\tau, a(t-1), T_{t:t+k}, P_{t:t+k}, H_{t:t+k}, d^p_{t:t+k}, d^q_{t:t+k}, \pi^p_{t:t+k}, \pi^f_{t:t+k}),$$

where $\tau = t/96$ is the time (normalized by number of 15 minute intervals in a day), $a(t-1)$ is the agent's previous action, $T_{t:t+k}$, $P_{t:t+k}$, and $H_{t:t+k}$ are current and k forecast steps of temperature,

pressure, and relative humidity, respectively, $d^p_{t:t+k}$ and $d^q_{t:t+k}$ are current and k forecast steps of electricity and steam demand, respectively, and $\pi^p_{t:t+k}$ and $\pi^f_{t:t+k}$ are current and k forecast steps of electricity and fuel price, respectively.

Action Space. The action space is a vector $a(t) \in \mathbb{R}^{15}$ specifying dispatch setpoints and other auxiliary variables for all turbines in the plant. Specifically, for each of three gas turbines, the agent specifies (a) a scalar turbine electricity output, (b) a scalar heat recovery steam flow, (c) a binary evaporative cooler switch setting, and (d) a binary power augmentation switch setting. In addition, for the steam turbine, the agent specifies (a) a scalar turbine electricity output, (b) a scalar steam flow through the plant condenser, and (c) an integer number of cooling tower bays employed.

Reward Function. The reward function is comprised of three components:

$$r(t) = -\left(r_f(a(t); T_t, P_t, H_t) + r_r(a(t); a(t-1)) + r_c(a(t); d^p_t, d^q_t)\right).$$

$r_f(a(t); T_t, P_t, H_t)$ is the generator fuel consumption in response to dispatch $a(t)$. $r_r(a(t); a(t-1))$ is the ramp cost, captured via an ℓ_1 norm penalty for any change in generator electricity dispatch between consecutive actions. $r_c(a(t); d^p_t, d^q_t)$ is a constraint violation penalty, penalizing any unmet electricity and steam demand, as well as any violation of the plant's dynamic operating constraints. The sum of these three components is negated to convert costs to rewards.

Distribution Shift. `CogenEnv` considers distribution shifts in the renewable generation profiles, and specifically, increasing penetration of wind energy. This increased variable renewable energy on the grid necessitates more frequent ramping in order to meet electricity demand, and may pose a challenge for RL algorithms trained on electricity demand traces without such variability.

Multiagent Setting. The multiagent setting treats each turbine unit (each of the three gas turbines and the steam turbine) as an individual agent whose action is the turbine's electricity dispatch decision and auxiliary variable settings. The negative reward of each agent is the sum of the corresponding turbine unit's fuel consumption, ramp cost, and dynamic operating constraint penalty, as well as a shared penalty for unmet electricity and steam demand that is split evenly across agents. All agents observe the global observation.

2.5 `BuildingEnv`

`BuildingEnv` considers the control of the heat flow in a multi-zone building so as to maintain a desired temperature setpoint. Building temperature simulation uses first-principled physics models. Users can either choose from a pre-defined list of buildings (Office small, School primary, Apartment midrise, and Office large) and three climate types and cities (San Diego, Tucson, New York) provided by the Department of Energy (DoE) Building Energy Codes Program [27] or define a customized `BuildingEnv` environment by importing any self-defined EnergyPlus building models. Each episode runs for 1 day, with 5-minute time intervals ($H = 288$, $\tau = 5/60$ hours).

Observation Space. For a building with M indoor zones, the state $s(t) \in \mathbb{R}^{M+4}$ contains observable properties of the building environment at timestep t:

$$s(t) = (T_1(t), \ldots, T_M(t), T_{\mathrm{E}}(t), T_{\mathrm{G}}(t), Q^{\mathrm{GHI}}(t), \bar{Q}^{\mathrm{p}}(t)),$$

where $T_i(t)$ is zone i's temperature at time step t, $\bar{Q}^{\mathrm{p}}(t)$ is the heat acquisition from occupant's activities, $Q^{\mathrm{GHI}}(t)$ is the heat gain from the solar irradiance, and $T_{\mathrm{G}}(t)$ and $T_{\mathrm{E}}(t)$ denote the ground and outdoor environment temperature. In practice, the agent may have access to all or part of the state variables for decision-making depending on the sensor setup. Note that the outdoor/ground temperature, room occupancy, and heat gain from solar radiance are time-varying uncontrolled variables from the environment.

Action Space. The action $a(t) \in [-1, 1]^M$ sets the controlled heating supplied to each of the M zones, scaled to $[-1, 1]$.

Reward Function. The objective is to reduce energy consumption while keeping the temperature within a given comfort range. The default reward function is a weighted ℓ_2 reward, defined as

$$r(t) = -(1-\beta)\left\|a(t)\right\|_2 - \beta\|T^{\mathrm{target}}(t) - T(t)\|_2$$

where $T^{\mathrm{target}}(t) = [T^{\mathrm{target}}_1(t), \ldots, T^{\mathrm{target}}_M(t)]^\top$ are the target temperatures and $T(t) = [T_1(t), \ldots, T_M(t)]^\top$ are the actual zonal temperatures. `BuildingEnv` also allows users to customize reward functions by changing the weight term β or the parameter p defining the ℓ_p norm.

Table 2: Distribution shift experiments

Environment	What shifts	Original setting	Shifted setting
`EVChargingEnv`	EV sessions, MOER	Summer 2019	Summer 2021
`DatacenterEnv`	MOER	May 2019	May 2021
`CogenEnv`	Wind penetration	0 MW wind	300 MW wind
`BuildingEnv`	Ambient temperature	Summer 2004	Winter 2003

Users can customize the reward function to consider CO_2 emissions and temperature constraints such as upper and lower temperature bounds.

Distribution Shift. `BuildingEnv` features distribution shifts in the ambient outdoor temperature profile T_{E} which varies with different seasons. `BuildingEnv` supports the distribution shifts due to the variation of seasons, located cities of the buildings, and can examine the challenges brought by such shifts in the RL environment.

Multiagent Setting. In the multiagent setting for `BuildingEnv`, each agent controls the heating action for a single zone in the building. It must coordinate with other agents to maximize overall reward. Each agent obtains the same global observation and reward.

3 Experiments

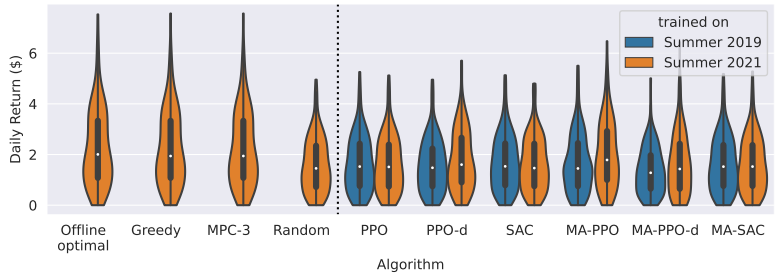
For each of the 5 environments in SustainGym, we implemented baseline non-RL algorithms as well as off-the-shelf RL algorithms trained using either RLLib [28] or Stable-Baselines3 (SB3) [29]. For most environments, we tested off-policy soft actor-critic (SAC) [30] and on-policy proximal policy optimization (PPO) [31]. Note that neither RLLib nor SB3 has an implementation of SAC that supports mixed discrete and continuous actions, as found in `CogenEnv`. For `EVChargingEnv`, we also tested multi-agent implementations of PPO and SAC, where the same policy is shared across agents. Non-RL algorithms tested include random policies and model predictive control (MPC), which is a model-based controller. Detailed descriptions of the implementations for each algorithm, including hyper-parameter tuning, are given in Appendix B. Finally, to test distribution shift, we trained RL agents in both "original" and "shifted" environments, then compared their performance on the shifted environment, as described in Table 2.

4 Discussion and Conclusion

Our experiments, shown in Figure 4, demonstrate a wide range of outcomes for off-the-shelf RL algorithms, with no single algorithm outperforming all the rest. In `EVChargingEnv`, for example, most of the RL algorithms perform no better than random actions, with the exception of multi-agent PPO with discrete actions. On `DatacenterEnv` and `BuildingEnv`, we notice a wider spread of returns across the different RL algorithms. In contrast, model-based MPC algorithms, where available, tend to perform more consistently than most RL algorithms.

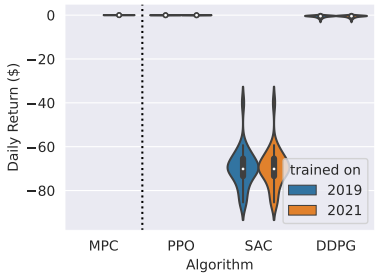
In terms of distribution shift, we see a wide range of outcomes between agents trained on the original environments versus the shifted environments. Surprisingly, in `CogenEnv`, both single-agent and multi-agent policies trained on the shifted environment perform worse on the shifted environment than agents trained on the original environment. We believe this result may be due to the increased variability of shifted environment, making the shifted environment harder to learn in. In `DatacenterEnv`, the shift in MOER values shows essentially no effect on agent performance. In `EVChargingEnv`, agents trained on the shifted environment generally perform slightly better than agents trained on the original environment. In `BuildingEnv`, agents trained on the shifted environment perform much better.

These results highlight that the distribution shifts present in SustainGym environments provide substantial opportunities for future research, including robust RL algorithms [32] as well as online learning under distribution shift. Developing RL algorithms that are robust to these natural distribution shifts will be critical for deploying RL in the real-world high-impact sustainability settings such as those modeled by SustainGym environments.

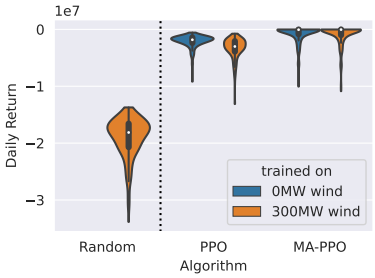


(a) `EVChargingEnv`

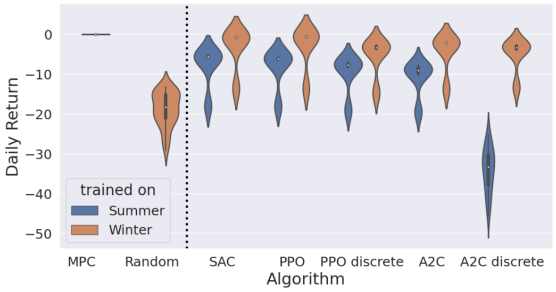
(b) `ElectricityMarketEnv`



(c) `DatacenterEnv`



(d) `CogenEnv`



(e) `BuildingEnv`

Figure 4: Experimental results for all 5 environments comparing performance on the shifted environment between RL algorithms trained on the original environment (blue) and RL algorithms trained on the shifted environment (orange). Policies using discretized actions are indicated with "-d", and multi-agent policies are prefixed with "MA-". For a complete description of the experiments, please see Appendix B.

Multi-agent RL. SustainGym is currently designed to support multi-agent RL in 3 environments, with the goal of upgrading all environments with multi-agent support in the future. In the two environments for which we tested multi-agent RL policies (`EVChargingEnv` and `CogenEnv`), the multi-agent PPO (MA-PPO) policies out-performed all other RL policies. We suspect that this may be because the action spaces in these environments factorize well across agents, so the multi-agent policies can learn more efficiently. Furthermore, we suspect that multi-agent policies may have the potential of performing better under distribution shift, since their environments are naturally non-stationary during training. We notice this to be true for both `EVChargingEnv` and `CogenEnv`: of the RL policies trained on the original environments, the multi-agent policies performed best when tested on the shifted environment.

Future work. SustainGym is under active development, with several key directions of future work:

- Comprehensive support for multi-agent RL. Currently, only 3 out of the 5 environments support multi-agent RL. We are working to extend the two other environments `ElectricityMarketEnv` and `DatacenterEnv` to the multi-agent RL setting. For `ElectricityMarketEnv`, we plan on introducing multiple batteries into the same transmission network, each controlled by separate competing agents. This will be the first

competitive multi-agent RL environment in SustainGym. (All other environments feature cooperative multi-agent RL.) For `DatacenterEnv`, we plan on introducing multiple datacenters spread across geographic regions to enable both temporal and geographic carbon-aware load shifting. Each datacenter would be its own agent.
- Different degrees of distribution shift. Currently, SustainGym environments feature a binary choice of distribution shift: an original environment, and a shifted environment. We plan on introducing more settings with varying degrees of distribution shift.
- More environments. We welcome new environment ideas and contributions to SustainGym and are working with potential collaborators to extend the scope of environments.

Limitations We conclude by acknowledging general limitations of SustainGym. First, SustainGym only captures very limited dimensions of sustainability (*i.e.*, energy and CO_2 emissions) and does not account for other aspects such as water usage and other pollutants associated with energy production. We welcome collaboration with experts in these other sustainability domains to help us improve the sustainability mission of SustainGym. Second, SustainGym is limited in the types of distribution shifts that are considered. Finally, while SustainGym environments have been designed to be reasonably representative of various sustainable energy settings, there is inevitably a gap between SustainGym simulations and actual hardware systems. A detailed discussion of the representativeness, generalizability, and limitations of each environment can be found in Appendix B.

Acknowledgments and Disclosure of Funding

We would like to acknowledge Steven Low, Tongxin Li, Zachary Lee, Lucien Werner, Zaiwei Chen, Ivan Jimenez, Pieter Van Santvliet, and Ameera Abdelaziz for their feedback and input during the preparation of SustainGym. The model underlying `CogenEnv` was developed in partnership with Beyond Limits and Enexsa. Funding for this project comes from a variety of sources, including NSF awards CNS-2146814, CPS-2136197, CNS-2106403, EPCN-2200692, and NGSDI-2105648; an NSF Graduate Research Fellowship; an Amazon AI4Science Fellowship; the Resnick Sustainability Institute; Hellman Fellowship; Amazon Web Services; and Beyond Limits. This work is partially supported by the Department of the Air Force under Air Force Contract No. FA8702-15-D-0001. Any opinions, findings, conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Department of the Air Force. Notwithstanding any copyright notice, U.S. Government rights in this work are defined by DFARS 252.227-7013 or DFARS 252.227-7014 as detailed above. Use of this work other than as specifically authorized by the U.S. Government may violate any copyrights that exist in this work.

References

[1] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–33, February 2015. ISSN 1476-4687. doi: 10.1038/nature14236. URL https://www.nature.com/articles/nature14236.

[2] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel, and Demis Hassabis. Mastering the game of Go without human knowledge. *Nature*, 550(7676):354–359, October 2017. ISSN 1476-4687. doi: 10.1038/nature24270. URL https://www.nature.com/articles/nature24270.

[3] Jens Kober, J. Andrew Bagnell, and Jan Peters. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11):1238–1274, September 2013. ISSN 0278-3649. doi: 10.1177/0278364913495721. URL https://doi.org/10.1177/0278364913495721.

[4] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. OpenAI Gym. *arXiv:1606.01540 [cs]*, June 2016. URL http://arxiv.org/abs/1606.01540.

[5] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 115(3):211–252, December 2015. ISSN 1573-1405. doi: 10.1007/s11263-015-0816-y. URL https://doi.org/10.1007/s11263-015-0816-y.

[6] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. In *International Conference on Learning Representations*, September 2018. URL https://openreview.net/forum?id=rJ4km2R5t7.

[7] Christopher Yeh, Chenlin Meng, Sherrie Wang, Anne Driscoll, Erik Rozi, Patrick Liu, Jihyeon Lee, Marshall Burke, David B. Lobell, and Stefano Ermon. SustainBench: Benchmarks for Monitoring the Sustainable Development Goals with Machine Learning. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 12 2021. doi: 10.48550/arXiv.2111.04724. URL https://openreview.net/forum?id=5HR3vCylqD.

[8] Sindhu Padakandla, Prabuchandran K. J, and Shalabh Bhatnagar. Reinforcement Learning in Non-Stationary Environments. *Applied Intelligence*, 50(11):3590–3606, November 2020. ISSN 0924-669X, 1573-7497. doi: 10.1007/s10489-020-01758-5. URL http://arxiv.org/abs/1905.03970.

[9] Elena Smirnova, Elvis Dohmatob, and Jérémie Mary. Distributionally Robust Reinforcement Learning, June 2019. URL http://arxiv.org/abs/1902.08708.

[10] Marcus Lapeyrolerie, Melissa S. Chapman, Kari E. A. Norman, and Carl Boettiger. Deep Reinforcement Learning for Conservation Decisions, June 2021. URL http://arxiv.org/abs/2106.08272.

[11] David Biagioni, Xiangyu Zhang, Dylan Wald, Deepthi Vaidhynathan, Rohit Chintala, Jennifer King, and Ahmed S. Zamzam. PowerGridworld: A Framework for Multi-Agent Reinforcement Learning in Power Systems. November 2021. doi: 10.48550/arXiv.2111.05969. URL http://arxiv.org/abs/2111.05969.

[12] Jose R. Vazquez-Canteli, Sourav Dey, Gregor Henze, and Zoltan Nagy. CityLearn: Standardizing Research in Multi-Agent Reinforcement Learning for Demand Response and Urban Energy Management, December 2020. URL http://arxiv.org/abs/2012.10504.

[13] Zachary J. Lee, Sunash Sharma, Daniel Johansson, and Steven H. Low. ACN-Sim: An Open-Source Simulator for Data-Driven Electric Vehicle Charging Research. *IEEE Transactions on Smart Grid*, 12(6):5113–5123, 11 2021. ISSN 1949-3061. doi: 10.1109/TSG.2021.3103156.

[14] Georgios Karatzinis, Christos Korkas, Michalis Terzopoulos, Christos Tsaknakis, Aliki Stefanopoulou, Iakovos Michailidis, and Elias Kosmatopoulos. Chargym: An EV Charging Station Model for Controller Benchmarking. IFIP Advances in Information and Communication Technology, pages 241–252, Cham, 2022. Springer International Publishing. ISBN 978-3-031-08341-9. doi: 10.1007/978-3-031-08341-9_20.

[15] Heba M. Abdullah, Adel Gastli, and Lazhar Ben-Brahim. Reinforcement Learning Based EV Charging Management Systems–A Review. *IEEE Access*, 9:41506–41531, 2021. ISSN 2169-3536. doi: 10.1109/ACCESS.2021.3064354.

[16] Hao Wang and Baosen Zhang. Energy Storage Arbitrage in Real-Time Markets via Reinforcement Learning. In *2018 IEEE Power Energy Society General Meeting (PESGM)*, pages 1–5, August 2018. doi: 10.1109/PESGM.2018.8586321.

[17] Hanchen Xu, Xiao Li, Xiangyu Zhang, and Junbo Zhang. Arbitrage of Energy Storage in Electricity Markets with Deep Reinforcement Learning, May 2019. URL `http://arxiv.org/abs/1904.12232`.

[18] Jun Cao, Dan Harrold, Zhong Fan, Thomas Morstyn, David Healey, and Kang Li. Deep Reinforcement Learning-Based Energy Storage Arbitrage With Accurate Lithium-Ion Battery Degradation Model. *IEEE Transactions on Smart Grid*, 11(5):4513–4521, September 2020. ISSN 1949-3061. doi: 10.1109/TSG.2020.2986333.

[19] Zachary J. Lee, Tongxin Li, and Steven H. Low. ACN-Data: Analysis and Applications of an Open EV Charging Dataset. In *Proceedings of the Tenth ACM International Conference on Future Energy Systems*, e-Energy '19, pages 139–149, New York, NY, USA, June 2019. Association for Computing Machinery. ISBN 978-1-4503-6671-7. doi: 10.1145/3307772.3328313. URL `https://doi.org/10.1145/3307772.3328313`.

[20] Zachary J. Lee, George Lee, Ted Lee, Cheng Jin, Rand Lee, Zhi Low, Daniel Chang, Christine Ortega, and Steven H. Low. Adaptive Charging Networks: A Framework for Smart Electric Vehicle Charging. *IEEE Transactions on Smart Grid*, 12(5):4339–4350, September 2021. ISSN 1949-3061. doi: 10.1109/TSG.2021.3074437. URL `https://ieeexplore.ieee.org/document/9409126`.

[21] Tongxin Li, Ruixiao Yang, Guannan Qu, Yiheng Lin, Steven Low, and Adam Wierman. Equipping Black-Box Policies with Model-Based Advice for Stable Nonlinear Control, June 2022. URL `http://arxiv.org/abs/2206.01341`.

[22] Probability Methods Subcommittee. Ieee reliability test system. *IEEE Transactions on Power Apparatus and Systems*, PAS-98(6):2047–2054, 1979. doi: 10.1109/TPAS.1979.319398.

[23] Clayton Barrows, Aaron Bloom, Ali Ehlen, Jussi Ikäheimo, Jennie Jorgenson, Dheepak Krishnamurthy, Jessica Lau, Brendan McBennett, Matthew O'Connell, Eugene Preston, Andrea Staid, Gord Stephen, and Jean-Paul Watson. The IEEE Reliability Test System: A Proposed 2019 Update. *IEEE Transactions on Power Systems*, 35(1):119–127, January 2020. ISSN 1558-0679. doi: 10.1109/TPWRS.2019.2925557.

[24] Abhishek Verma, Luis Pedrosa, Madhukar R. Korupolu, David Oppenheimer, Eric Tune, and John Wilkes. Large-scale cluster management at Google with Borg. In *Proceedings of the European Conference on Computer Systems (EuroSys)*, Bordeaux, France, 2015. URL `https://research.google/pubs/pub43438/`.

[25] Muhammad Tirmazi, Adam Barker, Nan Deng, Md Ehtesam Haque, Zhijing Gene Qin, Steven Hand, Mor Harchol-Balter, and John Wilkes. Borg: the Next Generation. In *EuroSys'20*, Heraklion, Crete, 2020. URL `https://research.google/pubs/pub49065/`.

[26] Ana Radovanovic, Ross Koningstein, Ian Schneider, Bokan Chen, Alexandre Duarte, Binz Roy, Diyue Xiao, Maya Haridasan, Patrick Hung, Nick Care, et al. Carbon-aware computing for datacenters. *IEEE Transactions on Power Systems*, 38(2):1270–1280, 2022.

[27] The Building Energy Codes Program. Prototype building models. URL https://www.energycodes.gov/prototype-building-models.

[28] Eric Liang, Richard Liaw, Robert Nishihara, Philipp Moritz, Roy Fox, Ken Goldberg, Joseph Gonzalez, Michael Jordan, and Ion Stoica. Rllib: Abstractions for distributed reinforcement learning. In *International Conference on Machine Learning*, pages 3053–3062. PMLR, 2018.

[29] Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-Baselines3: Reliable Reinforcement Learning Implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021. ISSN 1533-7928. URL http://jmlr.org/papers/v22/20-1364.html.

[30] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor, August 2018. URL http://arxiv.org/abs/1801.01290.

[31] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms, August 2017. URL http://arxiv.org/abs/1707.06347.

[32] Kishan Panaganti, Zaiyan Xu, Dileep Kalathil, and Mohammad Ghavamzadeh. Robust reinforcement learning using offline data. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 32211–32224. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/d01bda31bbcd780774ff15b534e03c40-Paper-Conference.pdf.

[33] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning, July 2019. URL http://arxiv.org/abs/1509.02971.

[34] Aled James and Daniel Schien. A low carbon kubernetes scheduler. In *6th International Conference on ICT for Sustainability, ICT4S 2019*, volume 2382 of *CEUR Workshop Proceedings*. CEUR-WS, June 2019.

[35] Nicolas Christianson, Christopher Yeh, Tongxin Li, Mahdi Torabi Rad, Azarang Golmohammadi, and Adam Wierman. Robustifying machine-learned algorithms for efficient grid operation. In *NeurIPS 2022 Workshop on Tackling Climate Change with Machine Learning*, 2022. URL https://www.climatechange.ai/papers/neurips2022/19.

[36] J. King, A. Clifton, and B. Hodge. Validation of Power Output for the WIND Toolkit. Technical Report NREL/TP-5D00-61714, 1159354, September 2014. URL http://www.osti.gov/servlets/purl/1159354/.

[37] CAISO. What the duck curve tells us about managing a green grid. Technical report, California Independent System Operator, November 2016.

[38] Chi Zhang, Yuanyuan Shi, and Yize Chen. BEAR: Physics-Principled Building Environment for Control and Reinforcement Learning. In *Proceedings of the Fourteenth ACM International Conference on Future Energy Systems*, e-Energy '23. arXiv, November 2022. doi: 10.48550/arXiv.2211.14744. URL http://arxiv.org/abs/2211.14744.

[39] US DOE. Engineering reference. https://energyplus.net/assets/nrel_custom/pdfs/pdfs_v22.1.0/EngineeringReference.pdf, 2022.

[40] Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous Methods for Deep Reinforcement Learning, June 2016. URL http://arxiv.org/abs/1602.01783.

Appendix

A Metadata

A.1 Hosting and maintenance

SustainGym is hosted as a Python package on GitHub and PyPI. SustainGym is semantically versioned, and the version accompanying this publication is designated v1.0.

A guide for contributors is available in the GitHub repo.

The following authors are the maintainers for each environment in SustainGym:

- Christopher Yeh: `EVChargingEnv`, `ElectricityMarketEnv`, `DatacenterEnv`
- Nicolas Christianson: `CogenEnv`
- Chi Zhang: `BuildingEnv`

A.2 Licenses and responsibility

SustainGym as a whole is released under a CC BY 4.0 license.[2] However, the `CogenEnv` and accompanying code is released under a more restrictive CC BY-NC-SA 4.0 license,[3] per the terms of the model provider Enexsa. These licenses are available in our GitHub repo.

The authors bear all responsibility in case of violation of rights. SustainGym does not contain any personally identifiable data, nor any offensive content.

A.3 Compute requirements

The experiments for this paper were run on Amazon AWS virtual machines, Google Colab notebooks, as well as a Caltech internal compute cluster. Most of these compute resources featured NVIDIA GPUs.

A.4 Intended uses

SustainGym is intended as a testbed for RL algorithms in sustainability-focused energy systems. While significant efforts have been undertaken to make the SustainGym environments closely match real-world systems, performance on SustainGym environments is not a guarantee of performance on real-world systems.

B Environment and Experiment Details

B.1 EVChargingEnv

Assumptions. In addition to the description given in Section 2.1, `EVChargingEnv` makes the following assumptions:

- EVs staying overnight can be ignored. Upon the start of each episode, we assume the garage is empty. Analysis of historical traces on the adaptive charging network show that at most 12 cars at one time stay through midnight. Because this number is small compared to the number of stations, we do not expect the assumption of an empty garage at the start of the day to significantly impact the accuracy of the environment in representing real-world settings.
- All EVs have identical batteries. Because ACN-Data does not include data on the model of each EV, yet ACN-Sim models battery charging dynamics which vary based on battery capacity, we assume that all EVs contain 100 kWh batteries and come with an initial state of charge such that if the EV were charged to the amount of the user-requested energy, the battery would be full. Energies requested over 100 kWh are capped at 100 kWh. We use the `Linear2StageBattery` battery model within ACN-Sim.

[2] `https://creativecommons.org/licenses/by/4.0/`

[3] `https://creativecommons.org/licenses/by-nc-sa/4.0/`

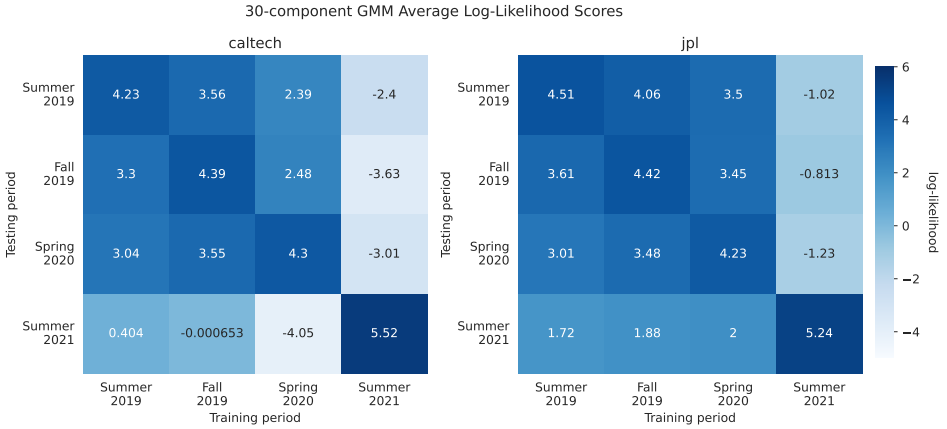


Figure 5: Log-likelihoods of data from testing periods (y-axis) for GMMs fitted on a training period (x-axis) using 30 components for the Caltech (left) and JPL (right) sites. A higher score implies a better fit. In all cases, GMMs scored highest on the period it was trained on. The significant drop in log-likelihood scores off-diagonal is indicative of distribution shift.

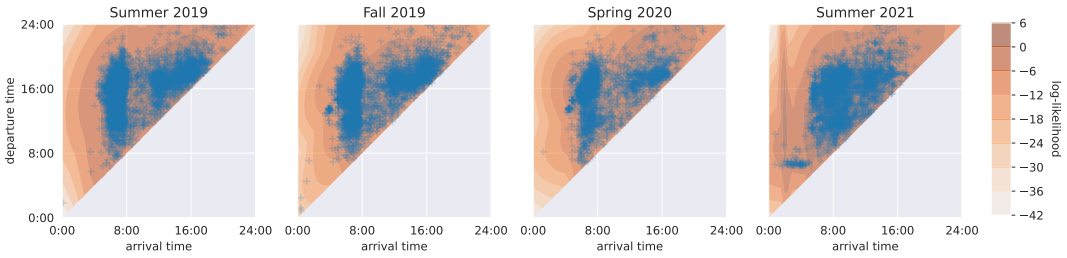


Figure 6: Like Figure 2, but for the JPL charging network.

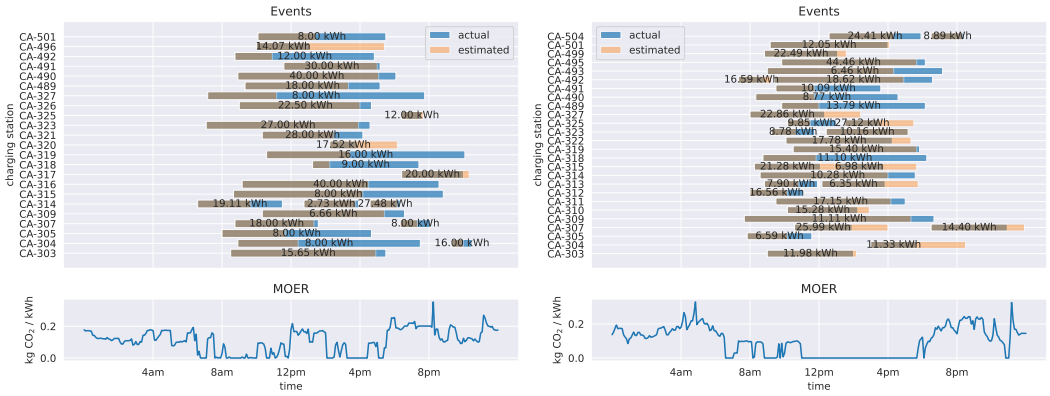


Figure 7: (left) A real full day's charging sessions from the Caltech charging network. (right) A randomly sampled trace of charging sessions from the GMMs trained on the Caltech historical data.

Thus, each historical charging session includes time of EV arrival, estimated departure, actual departure, energy delivered, and EVSE ID.

Feasible action space. Let $\mathcal{L}$ denote physical infrastructure resources (e.g., transformers or breakers) in the charging network. These infrastructure resources are characterized by (A, ϕ, c, v) where $A \in \mathbb{R}^{|\mathcal{L}| \times n}$ accounts for the charging network layout, $\phi \in \mathbb{R}^n$ is the voltage phase angle of each EVSE, $c \in \mathbb{R}^{|\mathcal{L}|}$ is the capacity limit for each resource, and $v \in \mathbb{R}_+$ is the EVSE voltage (in kV). Along with the demand at each EVSE, (A, ϕ, c) defines the time-dependent set of valid actions $\mathcal{A}_t$. Lastly, let $M = 32$ denote the maximum allowed pilot signal (in amps) for each EVSE.

When an agent gives the environment its desired normalized pilot signals $a(t) \in [0, 1]$, the projection into the convex hull of $\mathcal{A}_t$ is performed by solving the following convex optimization problem:

$$\begin{aligned}
\min_{\hat{a} \in \mathbb{R}^n} \quad & \|a(t) - \hat{a}\|_2 && && \text{(1a)} \\
\text{s.t.} \quad & 0 \leq \hat{a}_i \leq 1 && \forall i \in \{1, \ldots, n\} && \text{(1b)} \\
& M \hat{a}_i v \tau \leq e_i && \forall i \in \{1, \ldots, n\} && \text{(1c)} \\
& \underbrace{\left| \sum_{i=1}^{n} A_{li} M \hat{a}_i e^{j\phi_i} \right|}_{|I_l|} \leq c_l && \forall l \in \mathcal{L} && \text{(1d)}
\end{aligned}$$

Here, $j = \sqrt{-1}$ is the imaginary number and I_l is the aggregate current through constraint $l \in \mathcal{L}$. The quantity $M \hat{a}_i v \tau$ computes the energy (in kWh) to be charged from EVSE i during the next time step. The continuous pilot signals $M\hat{a}$ are then rounded to the set of discrete pilot signals supported by each EVSE, resulting in the final pilot signals $M\tilde{a}(t)$.

Reward function. The profit term $p(t)$ aims to maximize the amount of energy delivered to the EVs. Let $\pi \in \mathbb{R}_+$ denote a fixed marginal profit (in \$/kWh) that the EV charging network earns for each unit of energy delivered and $v \in \mathbb{R}_+$ be the voltage (in kV) of the charging network. Then,

$$p(t) = \pi \sum_{i=1}^{n} M \bar{a}_i v \tau = \pi M \bar{a}^\top \mathbf{1} v \tau$$

where $\mathbf{1}$ denotes a vector of all ones. The constraint violation cost $c_V(t)$ aims to reduce physical constraint violations and encourage the agent's action $a(t)$ to be in $\mathcal{A}_t$. We penalize violation costs with a weight λ_V (in \$/kWh), so

$$c_V(t) = \lambda_V \sum_{l \in \mathcal{L}} \max\{I_l(t) - c_l,\, 0\} v \tau$$

where I_l is the electrical current and c_l is the maximum current capacity at resource l. By default, we set $\lambda_V = 0.01$.

Finally, the CO_2 emissions cost $c_C(t)$ aims to reduce emissions by encouraging the agent to charge EVs when the MOER is low. We have

$$c_C(t) = P_{\text{CO}_2} m_t \sum_{i=1}^{n} M \bar{a}_i v \tau = P_{\text{CO}_2} m_t M \bar{a}^\top \mathbf{1} v \tau$$

Model predictive control (MPC). As a baseline non-RL algorithm, we consider a model predictive control (MPC) controller similar to what is proposed in [20]. Let $w \leq k$ denote the length of the lookahead window (up to the number of MOER forecast steps k). Then, at every time step t, the MPC controller solves the following optimization problem:

$$\max_{a^0,\ldots,a^{w-1}\in\mathbb{R}^n} \quad \sum_{k=0}^{w-1}(\pi - P_{\mathrm{CO}_2}\hat{m}_{t+k|t})Ma^{k\top}\mathbf{1}v\tau \tag{2a}$$

$$\text{s.t.} \quad 0 \preceq a^k \preceq 1 \qquad \forall k \in \{0,\ldots,w-1\} \tag{2b}$$

$$\sum_{k=0}^{w-1} Ma^k v\tau \preceq e \tag{2c}$$

$$a_i^k = 0 \qquad \forall i \in \{1,\ldots,n\},\ k \geq d_i \tag{2d}$$

$$\underbrace{\left|\sum_{i=1}^{n} A_{li}Ma_i^k e^{j\phi_i}\right|}_{|I_l|} \leq c_l, \qquad \forall l \in \mathcal{L},\ k \in \{0,\ldots,w-1\} \tag{2e}$$

The optimization problem plans pilot signals for the next w time steps to maximize revenue and minimize forecasted carbon emissions cost. (2c) ensures that the pilot signals do not over-charge the EVs. (2d) prevents charging EVs after their estimated departure times. (2e) is the usual infrastructure constraint.

Only the first planned pilot signal Ma^0 is used. It is rounded to the set of discrete pilot signals supported by each EVSE, resulting in the final pilot signal $M\tilde{a}(t)$. On the next time step, the MPC algorithm resolves the optimization problem. Figure 4a shows the distribution of returns for "MPC-3", which sets $w = 3$.

Training. The entire summer 2021 period [21] was selected as the testing period. We performed three splits: 1) by RL algorithm (PPO and multi-agent PPO vs. SAC and multi-agent SAC), 2) continuous vs. discrete action space, and 3) training data generated by GMMs based on out-of-distribution data (summer 2019) vs. in-distribution data (summer 2021). In each of the four cases, we used action projection during training and testing, eliminating costs of network violations. We tested 3 learning rates for each algorithm: PPO (5e-6, 5e-5, 5e-4) and SAC (1e-2, 1e-3, 1e-4). These learning rates were chosen as the default RLLib learning rates for each algorithm, scaled by factors of 10, 1, and 0.1. For each learning rate, three different random seeds were tested, and the model with the best training performance was selected.

Representativeness and Generalizability As mentioned in Section 2.1, `EVChargingEnv` is based on the actual EV charging networks in place at Caltech and the Jet Propulsion Laboratory (JPL), both located in Pasadena, California, U.S.A. `EVChargingEnv` uses a "digital twin" of these networks, called ACN-Sim [13], as well as real historical EV charging data [19], in the simulation of the RL environment. While different EV charging networks may have different constraints, the adaptive EV charging problem is similar across all networks (*i.e.*, deciding pilot signals for EVSEs to maximize energy delivery while minimizing costs and satisfying network and power constraints). In this way, `EVChargingEnv` is very representative of this type of control task. Furthermore, the code for ACN-Sim is open-source and well-documented,[4] which allows `EVChargingEnv` to be extended to model other charging networks as well.

Limitations As mentioned under "Assumptions" above, `EVChargingEnv` uses some simplifying assumptions, in part because of limited data availability. Furthermore, the reward function in `EVChargingEnv` currently assigns a fixed marginal profit per unit of energy delivered to an EV, whereas larger EV charging networks are often affected by time-varying wholesale electricity prices. Finally, electricity grid failures and communication failures are not modeled in `EVChargingEnv`.

B.2 ElectricityMarketEnv

System Model and Motivation. `ElectricityMarketEnv` aims to simulate grid-scale battery storage systems participating in an energy market for a regional transmission network. A battery has

[4] `https://acnportal.readthedocs.io/`

Table 3: `ElectricityMarketEnv` parameters. The k-th entry of a vector $\mathbf{g}_t$ is denoted $g_{k,t}$.

Param.	Domain	Unit	Description
t	$[1, \dots, T]$	5 min	index of real-time interval (usually 5-15 min)
T	$\mathbb{Z}_+$		number of periods in each epoch, typically 1 day for 5 min intervals
h	$\{0, 1, \dots, T\}$		length of the lookahead horizon
τ	$[t, \dots, t+h]$	5 min	index of the interval in each multi-interval lookahead subproblem
N	$\mathbb{Z}_+$		number of nodes in network
M	$\mathbb{Z}_+$		number of lines in the network
N_G	$\mathbb{Z}_+$		number of generators
N_D	$\mathbb{Z}_+$		number of loads
N_B	$\mathbb{Z}_+$		number of batteries
i	$[1, \dots, N]$		index of node
j	$[1, \dots, M]$		index of line
k	$[1, \dots, N_G, N_D, N_B]$		index of generator, load, or battery
B	$\mathbb{R}_+^{M \times M}$	Ω^{-1}	network susceptance matrix
C	$\{-1, 0, 1\}^{N \times M}$		node-line network incidence matrix
Σ	$\{-1, 0, 1\}^{N \times (N_D + N_G + 2N_B)}$		participant-to-node mapping matrix
H	$\mathbb{R}^{M \times N}$		generation shift factor matrix
$\mathbf{f}^{\max}$	$\mathbb{R}_+^M$	MVA	maximum power flow along each line
$\hat{\mathbf{d}}_t$	$\mathbb{R}^{N_D}$	MW	vector of predicted (average) load in interval t
$\underline{\mathbf{g}}, \overline{\mathbf{g}}$	$\mathbb{R}_+^{N_G}$	MW	min/max generation limits (assumed to be time invariant)
$\overline{\mathbf{b}^{\mathrm{c}}}, \overline{\mathbf{b}^{\mathrm{d}}}$	$\mathbb{R}_+^{N_B}$	MW	max charge/discharge limits (assumed to be time invariant)
$\underline{\mathbf{x}}, \overline{\mathbf{x}}$	$\mathbb{R}_+^{N_B}$	MWh	min/max state of charge limits (assumed to be time invariant)
$\mathbf{x}^{\mathrm{f}}$	$\mathbb{R}_+^{N_B}$	MWh	final state of charge required at interval T
$\mathbf{x}^0$	$\mathbb{R}_+^{N_B}$	MWh	initial state of charge at beginning of epoch
$\eta_k^{\mathrm{c}}, \eta_k^{\mathrm{d}}$	$(0, 1]$		charge and discharge efficiency of battery k
$c_{k,t}^{\mathrm{g}}$	$\mathbb{R}_+$	\$/MWh	marginal cost of generator k in interval t
$c_{k,t}^{\mathrm{b,c}}$	$\mathbb{R}_+$	\$/MWh	marginal cost of battery k charging in interval t
$c_{k,t}^{\mathrm{b,d}}$	$\mathbb{R}_+$	\$/MWh	marginal cost of battery k discharging in interval t

two objectives: make profit from price arbitrage (buy low, sell high) while minimizing its associated CO_2 emissions. The CO_2 emissions are accounted for when both charging and discharging:

- When a battery charges energy from the grid, it incurs the CO_2 emissions associated with grid's current marginal emissions rate (MOER).
- When a battery discharges energy into the grid, it offsets other generators that would have had to supply more electricity, thereby reducing CO_2 emissions commensurate with the grid's current MOER.

Network congestion refers to physical limits on transmission line capacity that prevent a generator from supplying electricity to a load at an arbitrary bus (a.k.a. "node") on the network. Such constraints lead to inefficiencies but are present in almost all real-world transmission networks. Thus, modeling these transmission constraints is important for ensuring electricity grid reliability.

The parameters of `ElectricityMarketEnv` are listed in Table 3. The default environment is based on the Region 1 in the IEEE Reliability Test System (RTS) [23], with 5-minute settlements ($\delta = 5/60$ hours), an episode length of $T = 288$ (1 day), and a lookahead horizon of $h = 36$ steps (3 hours). The IEEE RTS network features $N = 24$ nodes (a.k.a. buses) connected with $M = 38$ lines. Connected to the network are $N_G = 33$ conventional thermal generators and $N_D = 17$ loads. The network is slightly modified to add a single ($N_B = 1$) 80MWh battery system located at bus 11, with maximum charge/discharge rate $\overline{\mathbf{b}^{\mathrm{c}}} = \overline{\mathbf{b}^{\mathrm{d}}} = 20$MW, and initial and final state of charge (SOC)

$\underline{\mathbf{x}} = 0$, $\mathbf{x}^0 = \mathbf{x}^f = 40$MWh. Charge/discharge efficiency is set to $\eta^{\mathrm{c}} = \eta^{\mathrm{d}} = 0.95$, and the minimum generation limit for all generators is changed to $\underline{g} = 0$.

We assume that an independent system operator (ISO) solves standard security-constrained economic dispatch (SCED) with the DC (decoupled) power flow equations. "Security-constrained" means that line limits are respected. The formulation has the following features:

- 3 classes of market participants: generators, loads, and storage
- DC PF equations
- Line flows and constraints
- Linear cost functions
- Nodal demand timeseries

The formulation lacks the following features that may be considered important in a practical implementation of SCED:

- Unit commitment for thermal units
- Line and generator outage contingencies, *e.g.*, $N-1$ security
- Operating reserves
- Piecewise linear and quadratic generator cost functions
- Flexible, curtailable load
- Renewable generators
- Generator ramping constraints

Battery storage. There are N_B batteries in the system, of which only one is seeking to learn an optimal bidding strategy. Assign index $k = 1$ to this unit.

The bid must satisfy power bounds and non-simultaneity. The strategic battery must manage its own state of charge. The other $B-1$ batteries are assumed to be managed by the ISO. We do not consider the possibility of simultaneous charge/discharge.

The state of charge evolution of each battery is linear in charge/discharge efficiency parameters η_i^{c}, η_i^{d}. The charge/discharge limits and the SOC constraints are assumed to be known by the system operator for all units and are not part of the strategic agent's bid. The initial state of charge in each interval is given by the solution from the previous interval's lookahead optimization.

Generation and load. There are N_G generators in the system, with potentially multiple generators at each node. We do not consider ramping limits on generators or non-convexities from start-up and shutdown costs and constraints. The generators are constrained between their minimum ($\underline{\mathbf{g}} = 0$) and maximum production limits $\overline{\mathbf{g}}$. It is assumed that a feasible unit commitment exists for the given $\underline{\mathbf{g}}$ and $\overline{\mathbf{g}}$.

The load vector $\mathbf{d}_t \in \mathbb{R}^{N_D}$ is taken to be fixed (inflexible load). The demand prediction for $\tau > t$ is given by $\hat{\mathbf{d}}_\tau$, provided by an hourly day-ahead forecast.

Table 4: Definitions of problem variables. Variables are optimized in the economic dispatch problem.

Variable	Domain	Unit	Description
$\mathbf{p}_t$	$\mathbb{R}^N$	MW	vector of nodal power injections in interval t
$\mathbf{d}_t$	$\mathbb{R}^{N_D}$	MW	vector of load power injections in interval t
$\mathbf{g}_t$	$\mathbb{R}^{N_G}$	MW	vector of generator power injections in interval t
$\mathbf{b}_t^{\mathrm{c}}$	$\mathbb{R}_+^{N_B}$	MW	vector of battery charging power injections in interval t
$\mathbf{b}_t^{\mathrm{d}}$	$\mathbb{R}_+^{N_B}$	MW	vector of battery discharging power injections in interval t
$\mathbf{x}_t$	$\mathbb{R}_+^{N_B}$	MWh	vector of battery states of charge in interval t
λ_t	$\mathbb{R}$		Lagrange multiplier corresponding to the power balance constraint $\mathbf{1}^\top \mathbf{p}_t = 0$
$\boldsymbol{\mu}_t^{\pm}$	$\mathbb{R}^M$		Lagrange multiplier vectors corresponding to upper/lower line limit constraints
$\boldsymbol{\pi}_t$	$\mathbb{R}^N$	\$/MWh	market-clearing nodal price vector for interval t

Powerflow equations and line limits. The line susceptance matrix is $B = \mathrm{diag}(B_1, \ldots, B_M) \in \mathbb{R}_+^{M \times M}$ where $B_j \in \mathbb{R}$ is the susceptance of line j.

The matrix $C \in \{-1, 0, 1\}^{N \times M}$ is node-edge incidence matrix of the graph.

The slack bus of the network is assigned WLOG to node index $i = 1$. By convention the voltage angle of this node is fixed:

$$\theta_{1,t} = 0 \quad \forall t.$$

Given a vector of nodal real power injections $\mathbf{p}_t \in \mathbb{R}^N$, the power flows along lines in the network $\mathbf{f}_t \in \mathbb{R}^M$ is given by $\mathbf{f}_t = H\mathbf{p}_t$ with the generation shift matrix $H \in \mathbb{R}^{M \times N}$ defined as $H := BC^\top \left(CBC^\top\right)^\dagger$.

Economic dispatch problem. We implement a version of SCED called multi-interval lookahead real-time economic dispatch. In this version of the market clearing, the systems operator seeks to clear the market sequentially for each timestep (*e.g.*, every 5 mins), optimizing over the current interval t plus a lookahead horizon of h additional intervals. Only the solution from interval t is retained; the remaining h decisions are advisory. In systems with intertemporal constraints (*e.g.*, energy storage, unit commitment, ramping), it is necessary to perform this multi-interval dispatch to improve *ex-post* optimality and as well as to retain feasibility. In practice, North American ISOs all solve a version of multi-interval dispatch.

For interval t, the multi-interval SCED problem is

$$\begin{aligned}
\min_{\mathbf{g}_t, \mathbf{b}_t^{\mathrm{c}}, \mathbf{b}_t^{\mathrm{d}}} \quad & \delta \sum_{\tau=t}^{t+h} \left[\sum_{k=1}^{N_g} c_{k,\tau}^{\mathrm{g}} g_{k,\tau} + \sum_{k=1}^{N_b} c_{k,\tau}^{\mathrm{b,d}} b_{k,\tau}^{\mathrm{d}} - c_{k,\tau}^{\mathrm{b,c}} b_{k,\tau}^{\mathrm{c}} \right] && && \text{(3a)} \\
\text{s.t.} \quad & \Sigma[\mathbf{d}_\tau^\top, \mathbf{g}_\tau^\top, \mathbf{b}_\tau^{\mathrm{c}\top}, \mathbf{b}_\tau^{\mathrm{d}\top}]^\top = \mathbf{p}_\tau && \forall \tau = t, \ldots, t+h && \text{(3b)} \\
\lambda_\tau \ \perp \ & \mathbf{1}^\top \mathbf{p}_\tau \delta = 0 && \forall \tau = t, \ldots, t+h && \text{(3c)} \\
\boldsymbol{\mu}_\tau^+, \boldsymbol{\mu}_\tau^- \ \perp \ & |H\mathbf{p}_t| \leq \mathbf{f}^{\max} && \forall \tau = t, \ldots, t+h && \text{(3d)} \\
& \mathbf{d}_\tau = \hat{\mathbf{d}}_\tau && \forall \tau = t, \ldots, t+h && \text{(3e)} \\
& \underline{\mathbf{g}} \leq \mathbf{g}_\tau \leq \overline{\mathbf{g}} && \forall \tau = t, \ldots, t+h && \text{(3f)} \\
& \mathbf{0} \leq \mathbf{b}_\tau^{\mathrm{c}} \leq \overline{\mathbf{b}^{\mathrm{c}}} && \forall \tau = t, \ldots, t+h && \text{(3g)} \\
& \mathbf{0} \leq \mathbf{b}_\tau^{\mathrm{d}} \leq \overline{\mathbf{b}^{\mathrm{d}}} && \forall \tau = t, \ldots, t+h && \text{(3h)} \\
& \mathbf{x}_\tau = \mathbf{x}_{\tau-1} + \mathrm{diag}(\boldsymbol{\eta}^{\mathrm{c}}) \mathbf{b}_\tau^{\mathrm{c}} \delta - \mathrm{diag}(\boldsymbol{\eta}^{\mathrm{d}})^{-1} \mathbf{b}_\tau^{\mathrm{d}} \delta && \forall \tau = t, \ldots, t+h && \text{(3i)} \\
& \underline{\mathbf{x}} \leq \mathbf{x}_\tau \leq \overline{\mathbf{x}} && \forall \tau = t, \ldots, t+h && \text{(3j)} \\
& \mathbf{x}_{t-1} = \mathbf{x}_{t-1}^* && && \text{(3k)} \\
& \mathbf{x}_{t+h} = \mathbf{x}^{\mathrm{f}} && && \text{(3l)}
\end{aligned}$$

Here, $\mathbf{x}_{t-1}^*$ is the state of charge decision from the previous interval. The constraint says that the current state of charge is whatever the battery was charged/discharged to in the previous dispatch.

In order to ensure that the battery is never simultaneously charging and discharging (*i.e.*, for every $k \in \{1, \ldots, N_b\}$ and $\tau \in \{t, \ldots, t+h\}$, at most one of $b_{k,\tau}^{\mathrm{c}}$ and $b_{k,\tau}^{\mathrm{d}}$ should be nonzero), we can instead formulate the problem as a mixed-integer linear program:

$$
\begin{aligned}
\min_{\mathbf{g}_t, \mathbf{b}_t^{\mathrm{c}}, \mathbf{b}_t^{\mathrm{d}}} \quad & \delta \sum_{\tau=t}^{t+h} \left[\sum_{k=1}^{N_g} c_{k,\tau}^{\mathrm{g}} g_{k,\tau} + \sum_{k=1}^{N_b} c_{k,\tau}^{\mathrm{b,d}} b_{k,\tau}^{\mathrm{d}} - c_{k,\tau}^{\mathrm{b,c}} b_{k,\tau}^{\mathrm{c}} \right] && && \text{(4a)} \\
\text{s.t.} \quad & \Sigma[\mathbf{d}_\tau^\top, \mathbf{g}_\tau^\top, \mathbf{b}_\tau^{\mathrm{c}\top}, \mathbf{b}_\tau^{\mathrm{d}\top}]^\top = \mathbf{p}_\tau && \forall \tau = t, \ldots, t+h && \text{(4b)} \\
& \mathbf{1}^\top \mathbf{p}_\tau \delta = 0 && \forall \tau = t, \ldots, t+h && \text{(4c)} \\
& |H \mathbf{p}_t| \leq \mathbf{f}^{\max} && \forall \tau = t, \ldots, t+h && \text{(4d)} \\
& \mathbf{d}_\tau = \hat{\mathbf{d}}_\tau && \forall \tau = t, \ldots, t+h && \text{(4e)} \\
& \underline{\mathbf{g}} \leq \mathbf{g}_\tau \leq \overline{\mathbf{g}} && \forall \tau = t, \ldots, t+h && \text{(4f)} \\
& \mathbf{z}_\tau \in \{0,1\}^{N_b} && \forall \tau = t, \ldots, t+h && \text{(4g)} \\
& \mathbf{0} \leq \mathbf{b}_\tau^{\mathrm{c}} \leq \overline{\mathbf{b}^{\mathrm{c}}} \odot \mathbf{z}_\tau && \forall \tau = t, \ldots, t+h && \text{(4h)} \\
& \mathbf{0} \leq \mathbf{b}_\tau^{\mathrm{d}} \leq \overline{\mathbf{b}^{\mathrm{d}}} \odot (\mathbf{1} - \mathbf{z}_\tau) && \forall \tau = t, \ldots, t+h && \text{(4i)} \\
& \mathbf{x}_\tau = \mathbf{x}_{\tau-1} + \mathrm{diag}(\boldsymbol{\eta}^{\mathrm{c}}) \mathbf{b}_\tau^{\mathrm{c}} - \mathrm{diag}(\boldsymbol{\eta}^{\mathrm{d}})^{-1} \mathbf{b}_\tau^{\mathrm{d}} && \forall \tau = t, \ldots, t+h && \text{(4j)} \\
& \underline{\mathbf{x}} \leq \mathbf{x}_\tau \leq \overline{\mathbf{x}} && \forall \tau = t, \ldots, t+h && \text{(4k)} \\
& \mathbf{x}_{t-1} = \mathbf{x}_{t-1}^* && && \text{(4l)} \\
& \mathbf{x}_{t+h} = \mathbf{x}^{\mathrm{f}} && && \text{(4m)}
\end{aligned}
$$

However, the mixed-integer linear program does not produce dual-variables the same way that the original linear program does. In order to recover nodal prices, we first solve the mixed-integer linear program to determine the optimal values for $\mathbf{z}_\tau$. Then, in the linear program, we replace $\overline{\mathbf{b}^{\mathrm{c}}}$ with $\overline{\mathbf{b}^{\mathrm{c}}} \odot \mathbf{z}_\tau$ and $\overline{\mathbf{b}^{\mathrm{d}}}$ with $\overline{\mathbf{b}^{\mathrm{d}}} \odot (\mathbf{1} - \mathbf{z}_\tau)$, and we solve the linear program to get the nodal prices from the dual variables. Although this procedure requires solving two optimization problems (first the MILP, and later the linear program), in practice solving the linear program is very fast, since the optimization variables can be initialized to their optimal values from the MILP.

Market clearing and settlement. The market clears when optimization problem (3) has a feasible solution, denoted $(\mathbf{g}_t^*, \mathbf{b}_t^{\mathrm{c}*}, \mathbf{b}_t^{\mathrm{d}*})$. The nodal vector of market clearing prices $\boldsymbol{\pi} \in \mathbb{R}^N$ (in \$/MWh) is defined by a function of dual variables $\lambda_t^*, \boldsymbol{\mu}_t^{+*}, \boldsymbol{\mu}_t^{-*}$:

$$\boldsymbol{\pi}_t := \lambda_t^* \mathbf{1} + H^\top (\boldsymbol{\mu}_t^{+*} - \boldsymbol{\mu}_t^{-*})$$

For each interval t, the settlement rule for generator k at node i is:

1. Generator k produces power $g_{k,t}^*$
2. Generator k receives revenue $\pi_{i,t} g_{k,t}^*$

The settlement rules for loads and batteries are analogous. When a battery is charging ($b_{k,t}^{\mathrm{c}} > 0$), it pays the nodal price; when it is discharging ($b_{k,t}^{\mathrm{d}} > 0$), it receives the nodal price.

Representativeness and Generalizability While `ElectricityMarketEnv` is not modeled after any particular real-world transmission network, it simulates the transmission network from the widely-used IEEE Reliability Test System (IEEE RTS-24) [22]. As the IEEE RTS-24 test case did not include electricity load data, we chose to incorporate load data from the recent 2019 IEEE RTS-GMLC update [23], which was designed to be representative of a modern transmission network located in the southwestern U.S., featuring a variety of renewable and distributed generators as well as representative electricity load profiles. According to industry experts, both the IEEE RTS-24 and IEEE RTS-GMLC are simplified but standard test cases. Furthermore, `ElectricityMarketEnv` has a modular design and is readily modified to simulate a particular network.

The multi-time-step security-constrained economic dispatch problem (SCED) implemented in `ElectricityMarketEnv` (3) is also representative of how market operators schedule generators and determine nodal prices in most electricity markets in the U.S.A.

Limitations Designing `ElectricityMarketEnv` to be easy to use necessitated some limitations in what could be modeled. First, the default IEEE RTS-24 network only features 24 buses, which

is smaller than most real-world transmission networks. However, because each step of the environment requires solving a mixed-integer linear program, a larger test case would have significantly increased the amount of time taken in the environment and thus slowed down any RL training. Users who wish to test their RL algorithms in larger transmission networks are welcome to modify `ElectricityMarketEnv` for their needs.

Second, `ElectricityMarketEnv` is only based on data representative of modern electricity markets in the U.S.A., whereas many regions around the world still feature vertically-integrated energy monopolies and/or "traditional" electricity markets that may not necessarily solve a similar SCED optimization problem.

Finally, `ElectricityMarketEnv` only features distribution shifts in the form of changes in the marginal carbon emissions and load profiles between summer and winter months. In the real-world, distribution shifts also occur when more generators are added to the network, when older generators are retired, and when the transmission network changes (*e.g.*, when new transmission lines are added upgraded, or when transmission lines are taken offline by extreme weather events).

B.3 DatacenterEnv

System Model and Motivation. `DatacenterEnv` is inspired by the carbon-aware job scheduling approach adopted by Google's datacenters, as described in [26]. The Google approach can be summarized as follows. Each day, the system plans the 24-hour virtual capacity curve (VCC) for the next day by solving a constrained optimization problem whose objective is to minimize a weighted sum of expected carbon emissions and expected peak power consumption. The main constraint is that the VCC for the next day must sum to the 97th-percentile of the predicted total daily capacity requirement. Capacity is measured in terms of normalized CPU compute units. The VCC artificially limits the total datacenter capacity at each hour, forcing enqueued jobs to be scheduled later than they would otherwise run. The goal is to reduce the number of running jobs when the carbon intensity of the electrical grid (CO_2 emissions per electrical energy used) is high, and run more jobs when the carbon intensity is low. By time-shifting jobs, the datacenter is able to reduce its CO_2 emissions.

The VCC-based approach to carbon-aware job scheduling is *scheduler-agnostic* as it works with any scheduler (*e.g.*, a FIFO queue, or a priority queue). The details of job placement (which physical machine runs each job) and job prioritization (which jobs run first) are up to the scheduler.

Whereas the Google approach plans a whole day's VCC at once, `DatacenterEnv` allows RL agents to plan the VCC one hour at a time. Furthermore, `DatacenterEnv` makes the following simplifying assumptions compared to the Google approach described above:

- `DatacenterEnv` does not provide any predictions of the next-day predicted total daily capacity, instead relying on the reward function to penalize an agent for failing to plan a VCC that meets 97% of the day's capacity.
- `DatacenterEnv` does not account for peak power consumption.
- `DatacenterEnv` uses a simple scheduler based on a priority queue, and jobs are randomly placed on any available machine. Machine constraints are not accounted for.

In `DatacenterEnv`, the data used to simulate a datacenter workload is subsampled from Cluster A from the Google Cluster Workload Traces May 2019 dataset [25]. The 47,276 jobs in our subsample includes widely varying priorities, ranging from 0 (low priority) to 450 (high priority, latency-sensitive). We assume that our datacenter consists of 101 identical machines.

Training. We trained PPO, SAC, and deep deterministic policy gradient (DDPG) [33] RL algorithms on `DatacenterEnv` using PyTorch with RLLib's default hyperparameters for 60 episodes.

Representativeness, Generalizability, and Limitations `DatacenterEnv` is loosely based on a Google data center from May 2019. As most datacenters do not disclose their exact job scheduling mechanisms and machine specifications, `DatacenterEnv` uses several simplifying assumptions. It uses a simple priority queue for scheduling jobs, and it assumes that there are no constraints for which jobs can be placed on which machine. However, `DatacenterEnv` does use a subsample of actual job traces from a Google datacenter in May 2019, which includes information for each job such as priority, duration, and compute usage. As `DatacenterEnv` is specifically based on the VCC framework used by Google's approach to carbon-aware datacenters, it does not reflect efforts by other

cloud computing providers such as Microsoft [34] that may incorporate carbon emission estimates directly into the decision-making of their datacenter job schedulers.

Because Google only released datacenter job traces from May 2019, our ability to provide distribution shifts in `DatacenterEnv` over time is limited. We have thus chosen to only test shifts in the marginal carbon emissions rate, even though more realistic distribution shifts would also include changes in the statistics of datacenter jobs and in the capacities of the datacenter machines.

B.4 CogenEnv

System Model and Motivation. In the single-agent cogeneration environment, the agent controls a combined-cycle gas power plant with three gas turbines and one steam turbine. Gas power plants, similar to other conventional dispatchable generation types, suffer from loss of efficiency and degradation as a result of ramping energy generation up and down, thus necessitating the development of dispatch algorithms that can balance the tradeoff between fuel efficiency and ramp magnitude by anticipating future ramp needs while meeting demand in the highly constrained decision space. Such algorithms are even more crucial during the ongoing energy transition, as large quantities of variable and intermittent solar and wind resources are added to the grid. The variability of these resources requires dispatchable generators to ramp more frequently to balance supply with demand, and thus, dispatch algorithms that have been deployed historically may be suboptimal if they do not consider these ramp needs. The problem of dispatchable power plant operation in the face of uncertain renewable generation is thus an important problem, as better algorithms for generator dispatch will ensure that the performance and fuel efficiency of thermal generators will not degrade as renewables penetration increases.

The foundation of the system model in `CogenEnv` is a neural network that maps ambient conditions and dispatch variables to plant fuel consumption and other variables related to plant operation and constraints. This model was developed in partnership with Beyond Limits and Enexsa. A summary of the plant model inputs and outputs is provided in Table 5. In the table, "GT" abbreviates "Gas Turbine" and "HRSG" abbreviates "heat recovery steam generator". In brief, a generator dispatch decision includes generator outputs and steam flows for each of the three gas turbines, along with auxiliary binary decisions concerning the state of an evaporative cooler and power augmentation mode for the gas turbine. The dispatch decision also includes a generator output, condenser steam flow, and cooling tower bay count for the steam turbine. This dispatch decision, when provided to the plant model in conjunction with ambient temperature, pressure, and relative humidity (together comprising entries 1 through 18 in Table 5), yield a number of outputs (entries 19 through 47) detailing the fuel consumption of each turbine unit (entries 23-31) and of the plant as a whole (entry 22), the electric power generation of the plant (entry 19), the plant steam production (entry 20), and a number of dynamic operating limits on the dispatch variables (entries 32-47).

Action Space. The action space for the single-agent environment is a vector $a(t) \in \mathbb{R}^{15}$ specifying a dispatch decision for the cogeneration plant, and specifically, specifying values for entries 4 through 18 of Tables 5.

Observation Space. As described in the main body of the paper, observations take the form

$$s(t) = (\tau, a(t-1), T_{t:t+k}, P_{t:t+k}, H_{t:t+k}, d^p_{t:t+k}, d^q_{t:t+k}, \pi^p_{t:t+k}, \pi^f_{t:t+k}).$$

where τ is a normalized time – we consider each episode to be a single day, broken into 15 minute intervals, so $\tau = t/96$ – and $a(t-1)$ is the previous action. $T_{t:t+k}$, $P_{t:t+k}$, $H_{t:t+k}$, $d^p_{t:t+k}$, $d^q_{t:t+k}$, $\pi^p_{t:t+k}$ and $\pi^f_{t:t+k}$ are vectors of the current and k steps of future forecasts of temperature, pressure, relative humidity, electricity demand, steam demand, electricity price, and fuel price, respectively. The units and limits of temperature, pressure, and relative humidity are as in entries 1 through 3 of Table 5, electricity demand has units MW, and steam demand is in klb/h. While we do not use electricity and fuel prices in the reward for this environment, we include them in the observation to allow for modification of the agent reward to incorporate financial incentives; gas price data was obtained from the Henry Hub Natural Gas Spot Price dataset (`https://www.eia.gov/dnav/ng/hist/rngwhhdm.htm`) and electricity price data was obtained from historical day-ahead prices at the Houston zone of the ERCOT grid (`https://www.ercot.com/mp/data-products/data-product-details?id=NP4-180-ER`).

Table 5: Summary of input and output variables for `CogenEnv` plant model

Num.	Variable Description	Variable Group	Type	Feasible Region	Unit
1	Air Temperature	Ambient Conditions	Input	$[32, 115]$	°F
2	Air Pressure	Ambient Conditions	Input	$[14, 15]$	PSIA
3	Air Relative Humidity	Ambient Conditions	Input	$[0, 1]$	-
4	GT1 Generator Output	Turbine Block 1	Input	$[41.64, 168.3]$	MW
5	GT1 Evaporative Cooler Switch	Turbine Block 1	Input	$\{0, 1\}$	-
6	GT1 Power Augmentation Switch	Turbine Block 1	Input	$\{0, 1\}$	-
7	GT1 Steam Flow	Turbine Block 1	Input	$[403.2, 819.6]$	klb/h
8	GT2 Generator Output	Turbine Block 2	Input	$[41.49, 168.4]$	MW
9	GT2 Evaporative Cooler Switch	Turbine Block 2	Input	$\{0, 1\}$	-
10	GT2 Power Augmentation Switch	Turbine Block 2	Input	$\{0, 1\}$	-
11	GT2 Steam Flow	Turbine Block 2	Input	$[396.7, 817.4]$	klb/h
12	GT3 Generator Output	Turbine Block 3	Input	$[46.46, 172.4]$	MW
13	GT3 Evaporative Cooler Switch	Turbine Block 3	Input	$\{0, 1\}$	-
14	GT3 Power Augmentation Switch	Turbine Block 3	Input	$\{0, 1\}$	-
15	GT3 Steam Flow	Turbine Block 3	Input	$[439.0, 870.3]$	klb/h
16	Steam Generator Output	Steam Turbine	Input	$[25.65, 83.54]$	MW
17	Steam Flow through Condenser	Steam Turbine	Input	$[-1218, -318.1]$	klb/h
18	Number of Cooling Tower Bays	Steam Turbine	Input	$\{1, \ldots, 12\}$	-
19	Net Electric Power Output	Plant	Output	-	MW
20	Net Steam Export	Plant	Output	-	klb/h
21	Auxiliary Power Consumption	Plant	Output	-	MW
22	Total Fuel Consumption	Plant	Output	-	klb/h
23	GT1 Fuel Flow	Turbine Block 1	Output	-	klb/h
24	HRSG1 Fuel Flow	Turbine Block 1	Output	-	klb/h
25	Block 1 Total Fuel Consumption	Turbine Block 1	Output	-	klb/h
26	GT2 Fuel Flow	Turbine Block 2	Output	-	klb/h
27	HRSG2 Fuel Flow	Turbine Block 2	Output	-	klb/h
28	Block 2 Total Fuel Consumption	Turbine Block 2	Output	-	klb/h
29	GT3 Fuel Flow	Turbine Block 3	Output	-	klb/h
30	HRSG3 Fuel Flow	Turbine Block 3	Output	-	klb/h
31	Block 3 Total Fuel Consumption	Turbine Block 3	Output	-	klb/h
32	GT1 Min Power	Turbine Block 1	Output	-	MW
33	GT1 Max Power	Turbine Block 1	Output	-	MW
34	GT2 Min Power	Turbine Block 2	Output	-	MW
35	GT2 Max Power	Turbine Block 2	Output	-	MW
36	GT3 Min Power	Turbine Block 3	Output	-	MW
37	GT3 Max Power	Turbine Block 3	Output	-	MW
38	GT1 Min Steam	Turbine Block 1	Output	-	klb/h
39	GT1 Max Steam	Turbine Block 1	Output	-	klb/h
40	GT2 Min Steam	Turbine Block 2	Output	-	klb/h
41	GT2 Max Steam	Turbine Block 2	Output	-	klb/h
42	GT3 Min Steam	Turbine Block 3	Output	-	klb/h
43	GT3 Max Steam	Turbine Block 3	Output	-	klb/h
44	Steam Let-Down Flow Min Limit	Steam Turbine	Output	-	klb/h
45	Steam Let-Down Flow Max Limit	Steam Turbine	Output	-	klb/h
46	Steam Turbine Min Power	Steam Turbine	Output	-	MW
47	Steam Turbine Max Power	Steam Turbine	Output	-	MW

Reward Function The reward for the agent is defined as

$$r(t) = -\left(r_f(a(t); T_t, P_t, H_t) + r_r(a(t); a(t-1)) + r_c(a(t); d_t^p, d_t^q)\right).$$

The term $r_f(a(t); T_t, P_t, H_t)$ is the generator fuel consumption in response to dispatch decision $a(t)$; this is exactly the total fuel consumption of the plant (entry 22 of Table 5) resulting from model inputs $(T_t, P_t, H_t, a(t))$. The term $r_r(a(t); a(t-1))$ is the cost of ramping electricity generation up or down. Defining $a_j(t)$ to be the jth entry of Table 5 (so j ranges from 4 to 18 to include all entries comprising the action space), the ramp cost is defined as

$$\begin{aligned} r_r(a(t); a(t-1)) = \beta \cdot (&|a_4(t) - a_4(t-1)| + |a_8(t) - a_8(t-1)| \\ &+ |a_{12}(t) - a_{12}(t-1)| + |a_{16}(t) - a_{16}(t-1)|). \end{aligned}$$

In our experiments, we set the penalty magnitude $\beta = 2$, following [35]. The third term, $r_c(a(t); d_t^p, d_t^q)$, penalizes two types of constraint violation: the first is supply-demand imbalance. Let $x(t) \in \mathbb{R}^{47}$ be a vector containing plant model inputs and outputs as in Table 5; then $x_{19}(t)$ is the total power output of the plant and $x_{20}(t)$ is the total steam output of the plant. The form of the penalty on supply-demand imbalance is as follows:

$$r_c^{\text{sd}}(a(t); d_t^p, d_t^q) = \gamma \cdot \left(\max\{0, d_t^p - x_{19}(t)\} + \max\{0, d_t^q - x_{20}(t)\}\right).$$

The second type of constraint violation penalized is that of the dynamic operating constraints. These are determined by the plant outputs in entries 32 through 47 of Table 5, and constrain dispatch variables to lie within certain intervals. This penalty takes the following form:

$$\begin{aligned} r_c^{\text{dyn}}(a(t); d_t^p, d_t^q) = \gamma \cdot \big(& \max\{0, x_4(t) - x_{32}(t)\} + \max\{0, x_{33}(t) - x_4(t)\} \\ & + \max\{0, x_8(t) - x_{34}(t)\} + \max\{0, x_{35}(t) - x_8(t)\} \\ & + \max\{0, x_{12}(t) - x_{36}(t) + \max\{0, x_{37}(t) - x_{12}(t)\} \\ & + \max\{0, x_7(t) - x_{38}(t)\} + \max\{0, x_{39}(t) - x_7(t)\} \\ & + \max\{0, x_{11}(t) - x_{40}(t)\} + \max\{0, x_{41}(t) - x_{11}(t)\} \\ & + \max\{0, x_{15}(t) - x_{42}(t)\} + \max\{0, x_{43}(t) - x_{15}(t)\} \\ & + \max\{0, x_{17}(t) - x_{44}(t)\} + \max\{0, x_{17}(t) - x_{45}(t)\} \\ & + \max\{0, x_{16}(t) - x_{46}(t)\} + \max\{0, x_{47}(t) - x_{16}(t)\}\big). \end{aligned}$$

In our experiments, we set the penalty magnitude parameter $\gamma = 1000$. The total constraint violation penalty is simply the sum of the supply-demand violation penalty $r_c^{\text{sd}}(a(t); d_t^p, d_t^q)$ and the dynamic operating constraint penalty $r_c^{\text{dyn}}(a(t); d_t^p, d_t^q)$.

Distribution Shift. In our distribution shift scenario, we consider the addition of 300 MW of wind energy onto the grid, which increases variability of net energy demand and changes the shape of load profiles. We obtain simulated wind speed profiles using the WIND Toolkit [36] at an altitude of 100m at $(39.970406, -128.77481)$ at 15 minute intervals, and transform these into electricity generation profiles using the power curve for an IEC Class 2 turbine, scaling to obtain a maximum generation level of 300 MW.

Multi-Agent Setting. In the multi-agent version of the environment, each of the four agents represents one of the blocks of the cogeneration plant: the first three control the three gas turbine blocks, and the fourth controls the steam generation block. Each agent observes the global observation, but controls only the variables relevant for its block — thus, agent 1 controls variables 4 through 7 in Table 5, agent 2 controls variables 8 through 11, agent 3 controls variables 12 through 15, and agent 4 controls variables 16 through 18. Each agent's reward only includes the terms relevant for its unit - thus, for instance, agent 1 pays fuel cost according to entry 25, ramp cost $\beta \cdot (|a_4(t) - a_4(t-1)|)$, and dynamic operating constraint penalty $\gamma \cdot \big(\max\{0, x_4(t) - x_{32}(t)\} + \max\{0, x_{33}(t) - x_4(t)\} + \max\{0, x_7(t) - x_{38}(t)\} + \max\{0, x_{39}(t) - x_7(t)\}\big)$. However, the supply-demand imbalance constraint penalty is shared equally amongst all agents: each pays $\frac{1}{4} r_c^{\text{sd}}(a(t); d_t^p, d_t^q)$.

Training. We train and test the performance of several algorithms on 250 days of data (ambient conditions, demands, and prices) between May 2021 and January 2022. For the testing environment, we use the scenario with 300 MW of wind generation. We performed two splits: 1) by algorithm (PPO vs. random actions), and 2) training data with out-of-distribution data (300 MW wind generation) vs. in-distribution data (no wind generation). We tested 3 learning rates for PPO and multi-agent PPO: 5e-6, 5e-5, 5e-4. For each learning rate, three different random seeds were tested, and the model with the best training performance was selected.

Representativeness, Generalizability, and Limitations `CogenEnv` is based on an actual combined-cycle power plant operated in the U.S.A. The general configuration has remained unchanged, but certain parameters such as generation capacity of units have been changed to protect the original data. More specifically, the general configuration of gas turbines, steam turbine, cooling tower, and binary setpoints are the same. Temperature and pressure settings of intermediate pressure and high pressure steams are also the same. However, some design parameters such as power and steam generation capacities of each unit and units' efficiencies has been scaled/shifted.

According to our collaborator Mehdi Hosseini at Beyond Limits: "Gas and steam turbines are generally modeled by simulating the Brayton and Rankine thermodynamic cycles. Besides the considered setpoints, some gas turbines might include extra minor setpoints like switching the duct burner on and off, which is considered always on in this model. The behavior of the steam turbine is more complex and interrelated with the structure of the condenser and cooling tower. In this simulation, a forced air cooling tower structure is employed, a typical method in combined cycle power plants. It's worth noting that using a different cooling system might influence the efficiency and generation limits of the steam turbine. In summary, the model accurately represents single or multi-gas turbine power plants, as well as combined-cycle power plants with a forced air cooling tower."

Finally, the distribution shift modeled in `CogenEnv` comes from changes in external renewable wind energy penetration, which causes greater need for ramping of fuel-based generators, leading to higher ramping costs. The is reflective of actual changes and challenges occurring in modern electricity grids, often referred to as the "duck curve problem" [37]. However, `CogenEnv` currently does not model other sources of distribution shift such as changing fuel prices.

B.5 BuildingEnv

Table 6: Definitions of `BuildingEnv` variables.

Variable	Domain	Unit	Description
$T_i(t)$	$\mathbb{R}$	°C	temperature in zone i at time t
$T_{\mathrm{G}}(t)$	$\mathbb{R}$	°C	ground temperature at time t
$T_{\mathrm{E}}(t)$	$\mathbb{R}$	°C	outdoor/ambient temperature at time t
$Q^{\mathrm{GHI}}(t)$	$\mathbb{R}_+$	W/m^2	heat gain from solar irradiance per square meter at time t ("GHI" is an acronym for "global horizontal irradiance")
$Q_i^{\mathrm{s}}(t)$	$\mathbb{R}_+$	W	heat acquisition from solar irradiance in zone i at time t
$Q_i^{\mathrm{p}}(t)$	$\mathbb{R}_+$	W	heat acquisition from occupants' activities in zone i at time t
$Q_i^{\mathrm{h}}(t)$	$\mathbb{R}$	W	heat acquisition from HVAC in zone i at time t
$N_i(t)$	$\mathbb{N}$		number of occupants in zone i at time t
A_i^{win}	$\mathbb{R}_+$	m^2	window area for zone i
$Q_i^{\mathrm{h,max}}$	$\mathbb{R}_+$	W	maximum heat acquisition from HVAC in zone i
$a_i(t)$	$[-1, 1]$		action value, normalized heat acquisition from HVAC in zone i at time t ($+$ for heating, $-$ for cooling)
w_i	$[0, 1]$		efficiency coefficient for HVAC in zone i

Observation space. `BuildingEnv` considers a building with M indoor zones. The observation $s(t) \in \mathbb{R}^{M+4}$ is the concatenation of the zonal observations $T(t) \in \mathbb{R}^M$ and the environmental observations $s^{\mathrm{env}}(t) \in \mathbb{R}^4$:

$$
\begin{aligned}
s(t) &= [T(t)^\top, s^{\mathrm{env}}(t)^\top]^\top \\
T(t) &= [T_1(t), \ldots, T_M(t)]^\top \\
s^{\mathrm{env}}(t) &= [T_{\mathrm{E}}(t), T_{\mathrm{G}}(t), Q^{\mathrm{GHI}}(t), \bar{Q}^{\mathrm{p}}(t)]^\top.
\end{aligned}
$$

We split the observation $s(t)$ into the two components to emphasize that $T(t)$ is affected by the agent's control actions, whereas $s^{\mathrm{env}}(t)$ is the set of exogenous time-varying environmental variables that are unaffected by the agent's control actions. The descriptions for each of these variables can be found in Table 6.

Action space. The action $a(t) = [a_1(t), \ldots, a_M(t)]^\top \in [-1, 1]^M$ is a set of controllable actions for building heat control, where $a_i(t)$ is the controlled heating supplied to zone i. We normalize the HVAC power consumption with respect to the maximum heating capacity, so that $a_i(t)$ is bounded in $[-1, 1]$. The resulting heat supply from the HVAC system to zone i at time t is given by

$$
Q_i^{\mathrm{h}}(t) = a_i w_i Q_i^{\mathrm{h,max}},
$$

where w_i is the efficiency coefficient for the HVAC system in zone i and $Q_i^{\mathrm{h,max}}$ is the maximum heat supply from the HVAC in zone i.

Given an action $a(t)$, the `BuildingEnv.step()` method simulates the next state via the physics-based state transition model described below.

System model. The building simulation builds upon the physics-principled environment in [38], which includes a reduced linear Resistance-Capacitance (RC) model for heat transfer with nonlinear residual modeling for occupants' activities and solar irradiance. The zonal thermal dynamics are given by

$$C_i \frac{dT_i}{dt} = \sum_{j \in \mathcal{N}(i)} \frac{T_j - T_i}{R_{i,j}} + Q_i^{\mathrm{h}} + Q_i^{\mathrm{s}} + Q_i^{\mathrm{p}}, \tag{5}$$

where $\mathcal{N}(i)$ is the set of zones neighboring zone i. That is,

$$\mathcal{N}(i) = \{j \in \{1, \ldots, M, \mathrm{G}, \mathrm{E}\} \mid j \neq i, \text{ zone } j \text{ shares a wall with zone } i\}.$$

The set $\mathcal{N}(i)$ may include G and E, if zone i is on the ground floor or connected to the outside environment, respectively. C_i is the thermal capacitance (in J/K), and $R_{i,j} = R_{j,i}$ is the thermal resistance (in K/W) between zones i, j. If zone i and j are not neighboring zones (*i.e.*, $j \notin \mathcal{N}(i)$), we set $R_{i,j} = R_{j,i} = +\infty$. Q_i^{h} is the controlled heating/cooling flow distributed to each zone as described above. Q_i^{s} and Q_i^{p} adhere to models outlined in the EnergyPlus documentation [39], which represent the heat accumulation in zone i from solar heat acquisition from windows and indoor human activities, respectively.

To capture the solar heat acquisition from windows, let α_{GHI} denote the coefficient determining the solar heat gain for windows, A_i^{win} be the window area for zone i (in m^2), and $Q^{\mathrm{GHI}}(t)$ be the heat gain from global horizontal irradiance (in $\mathrm{W/m}^2$). Then the accumulated solar heat from windows at time t can be calculated as

$$Q_i^{\mathrm{s}}(t) = \alpha_{\mathrm{GHI}} \cdot A_i^{\mathrm{win}} \cdot Q^{\mathrm{GHI}}(t).$$

To calculate heat gain originating from human activities at time t, $Q_i^{\mathrm{p}}(t)$, we consider N_i to symbolize the population in zone i, and $\bar{Q}^{\mathrm{p}}(t)$ signifies the discernible heat contributed by a single individual's activities. We assume that the number of people in each zone does not change over the course of the simulation, and we leave the task of modeling time-varying population for future work. Then,

$$Q_i^{\mathrm{p}}(t) = \bar{Q}^{\mathrm{p}}(t) \cdot N_i.$$

The computation of sensible heat per individual denoted as $\bar{Q}_p(t)$, is given by a polynomial function from the EnergyPlus documentation [39, p.1299],

$$\bar{Q}^{\mathrm{p}}(t) = c_1 + c_2 m_t + c_3 m_t^2 + c_4 \bar{T}(t) - c_5 m_t \bar{T}(t) + c_6 m_t^2 \bar{T}(t) - c_7 \bar{T}^2(t) + c_8 m_t \bar{T}^2(t) - c_9 m_t^2 \bar{T}^2(t),$$

where m_t signifies the population metabolic rate (in W) at time t, $\bar{T}(t) = \frac{1}{M}\sum_{i=1}^{M} T_i(t)$ is the average zone temperature, and $c_1, \ldots, c_9$ are constants that have been deduced by fitting sensible heat data under a variety of conditions. Both $Q_i^{\mathrm{s}}(t)$ and $Q_i^{\mathrm{p}}(t)$ fluctuate over time and embody heat emanations from the environment and occupants' activities that are beyond control.

The state evolution in (5), together with the definitions of $Q_i^{\mathrm{h}}, Q_i^{\mathrm{s}}, Q_i^{\mathrm{p}}$ can be written as

$$\dot{T}(t) = AT(t) + Bu(t) + Df(T(t)), \tag{6}$$

where $u(t) = [T_{\mathrm{G}}(t), T_{\mathrm{E}}(t), a_1(t), \ldots, a_M(t), Q^{\mathrm{GHI}}(t)]^\top$. Let the indicator variables $\mathbf{1}_{\mathrm{G},i} := \mathbf{1}[\mathrm{G} \in \mathcal{N}(i)]$ and $\mathbf{1}_{\mathrm{E},i} := \mathbf{1}[\mathrm{E} \in \mathcal{N}(i)]$ encode zone i's connectivity to the ground and the outside environment, respectively. Then the A and B matrices are

$$A = \begin{bmatrix} \sum_{j \in \mathcal{N}(i)} \frac{-1}{C_1 R_{1,j}} + \frac{c_p N_1}{M C_1} & \frac{1}{C_1 R_{1,2}} + \frac{c_p N_1}{M C_1} & \cdots & \frac{1}{C_1 R_{1,M}} + \frac{c_p N_1}{M C_1} \\ \frac{1}{C_2 R_{2,1}} + \frac{c_p N_2}{M C_2} & \sum_{j \in \mathcal{N}(i)} \frac{-1}{C_2 R_{2,j}} + \frac{c_p N_2}{M C_2} & \ddots & \vdots \\ \vdots & & \ddots & \frac{1}{C_{M-1} R_{M-1,M}} + \frac{c_p N_{M-1}}{M C_{M-1}} \\ \frac{1}{C_M R_{M,1}} + \frac{c_p N_M}{M C_M} & \cdots & \cdots & \sum_{j \in \mathcal{N}(i)} \frac{-1}{C_M R_{M,j}} + \frac{c_p N_M}{M C_M} \end{bmatrix},$$

$$B = \begin{bmatrix} \frac{\mathbf{1}_{\mathrm{G},1}}{C_1 R_{\mathrm{G},1}} & \frac{\mathbf{1}_{\mathrm{E},1}}{C_1 R_{\mathrm{E},1}} & \frac{w_1 Q_1^{\mathrm{h,max}}}{C_1} & 0 & \cdots & 0 & \frac{\alpha_{\mathrm{GHI}} A_1^{\mathrm{win}}}{C_1} \\ \frac{\mathbf{1}_{\mathrm{G},2}}{C_2 R_{\mathrm{G},2}} & \frac{\mathbf{1}_{\mathrm{E},2}}{C_2 R_{\mathrm{E},2}} & 0 & \frac{w_2 Q_2^{\mathrm{h,max}}}{C_2} & \cdots & 0 & \frac{\alpha_{\mathrm{GHI}} A_2^{\mathrm{win}}}{C_2} \\ \vdots & \vdots & \vdots & & \ddots & & \vdots \\ \frac{\mathbf{1}_{\mathrm{G},M}}{C_M R_{\mathrm{G},M}} & \frac{\mathbf{1}_{\mathrm{E},M}}{C_M R_{\mathrm{E},M}} & 0 & 0 & \cdots & \frac{w_M Q_M^{\mathrm{h,max}}}{C_M} & \frac{\alpha_{\mathrm{GHI}} A_M^{\mathrm{win}}}{C_M} \end{bmatrix}.$$

The vector D in Eq (6) is defined as $D = \left[\frac{N_1}{C_1}, \frac{N_2}{C_2}, \cdots, \frac{N_M}{C_M}\right]^\top$, and the nonlinear function for sensible heat calculation is

$$\begin{aligned} f(T(t)) &:= \bar{Q}^{\mathrm{p}}(t) - c_4\bar{T}(t) \\ &= c_1 + c_2 m_t + c_3 m_t^2 - c_5 m_t \bar{T}(t) + c_6 m_t^2 \bar{T}(t) - c_7 \bar{T}^2(t) + c_8 m_t \bar{T}^2(t) - c_9 m_t^2 \bar{T}^2(t). \end{aligned}$$

We convert the continuous-time system model into a discrete-time model,

$$T[k+1] = \bar{A}T[k] + \bar{B}u[k] + D\bar{f}[k] \tag{7}$$

where the discrete-time system matrices $\bar{A}, \bar{B}, \bar{f}$ are obtained from the continuous-time model parameters $A, B, f(T(t))$ using zero-order hold method with discretization time ΔT. In particular, $\bar{A} = e^{A\Delta T}$, $\bar{B} = A^{-1}(\bar{A} - I)B$, and $\bar{f}[k] = \int_{k\Delta T}^{(k+1)\Delta T} f(T(\tau))\,\mathrm{d}\tau$.

Reward function. The main objective of building control is to reduce energy consumption while keeping the temperature within a given comfort range. Therefore, the reward function penalizes both temperature deviations and HVAC energy consumption:

$$r(t) = -(1-\beta)\left\|a(t)\right\|_p - \beta\|T^{\mathrm{target}}(t) - T(t)\|_p\,,$$

where $T^{\mathrm{target}}(t) = [T_1^{\mathrm{target}}(t), \ldots, T_M^{\mathrm{target}}(t)]^\top$ are the target temperatures. The parameters β and p are user-customizable scalars, where β trades off between the energy consumption and temperature deviation penalties, and p determines the norm used. An important future direction is to incorporate CO_2 emissions into consideration in the default reward function for building control environment.

Building and weather types for simulation. The prototype buildings included in `BuildingEnv` are derived from the Department of Energy (DOE) Commercial Reference Building Models. The models include 16 commercial building types in 19 locations. Users can download all models at `https://www.energycodes.gov/prototype-building-models`. Since `BuildingEnv` is compatible with EnergyPlus, users could also create their own model in the EnergyPlus editor and load the generated table file into the `BuildingEnv`.

- *Available building types*: ApartmentHighRise, ApartmentMidRise, Hospital, HotelLarge, HotelSmall, OfficeLarge, OfficeMedium, OfficeSmall, OutPatientHealthCare, RestaurantFastFood, RestaurantSitDown, RetailStandalone, RetailStripmall, SchoolPrimary, SchoolSecondar, Warehouse.
- *Available cities and weather types*: Ho Chi Minh City (Extremely Hot Humid), Dubai (Extremely Hot Dry), Honolulu (Very Hot Humid), New Delhi (Very Hot Dry), Tampa (Hot Humid), Tucson (Hot Dry), Atlanta (Warm Humid), El Paso (Warm Dry), San Diego (Warm Marine), New York (Mixed Humid), Albuquerque (Mixed Dry), Seattle (Mixed Marine), Buffalo (Cool Humid), Denver (Cool Dry), Port Angeles (Cool Marine), Rochester (Cold Humid), Great Falls (Cold Dry), International Falls (Very Cold), Fairbanks (Subarctic/Arctic).

Example. A practical usage example is demonstrated in Figure 8. The intent of this code excerpt is to imitate an "OfficeLarge" type structure in San Diego, with "Warm Marine" weather conditions, employing arbitrarily selected actions. Upon the creation of a building model, comprehensive zone information is then displayed as shown in Figure 8. At each control time step, the RL agent observes the present state $s(t)$ and produces a corresponding action $a(t)$. The environment `BuildingEnv`, in turn, incorporates this action and integrates it into the building state-space model to project the subsequent state $s(t+1)$ for the approaching time step. Each episode runs for 1 day, with 5-minute time intervals ($H = 288$, $\tau = 5/60$ hours). The agent controls the supplied heat flow to each zone and is rewarded for maintaining the desired temperature at the minimum electricity usage. Figure 9 visualizes the indoor temperature in different zones of OfficeLarge for 1 day without control, initialized at $13.2°C$.

Model predictive control (MPC). As a baseline non-RL algorithm, we consider a model predictive control (MPC) controller similar to the `EVChargingEnv` environment. At every time step t, with a

```
>>> from sustaingym.envs.building import BuildingEnv, ParameterGenerator
>>> params = ParameterGenerator(building='OfficeLarge', weather='Warm_Marine', location='SanDiego')
>>> env = BuildingEnv(params)

###############All Zones from Ground############
BASEMENT  [Zone index]:  0
DATACENTER_BASEMENT_ZN_6  [Zone index]:  1
CORE_BOTTOM  [Zone index]:  2
PERIMETER_BOT_ZN_3  [Zone index]:  3
PERIMETER_BOT_ZN_2  [Zone index]:  4
PERIMETER_BOT_ZN_1  [Zone index]:  5
PERIMETER_BOT_ZN_4  [Zone index]:  6
DATACENTER_BOT_ZN_6  [Zone index]:  7
GROUNDFLOOR_PLENUM  [Zone index]:  8
CORE_MID  [Zone index]:  9
PERIMETER_MID_ZN_3  [Zone index]:  10
PERIMETER_MID_ZN_2  [Zone index]:  11
PERIMETER_MID_ZN_1  [Zone index]:  12
PERIMETER_MID_ZN_4  [Zone index]:  13
DATACENTER_MID_ZN_6  [Zone index]:  14
MIDFLOOR_PLENUM  [Zone index]:  15
CORE_TOP  [Zone index]:  16
PERIMETER_TOP_ZN_3  [Zone index]:  17
PERIMETER_TOP_ZN_2  [Zone index]:  18
PERIMETER_TOP_ZN_1  [Zone index]:  19
PERIMETER_TOP_ZN_4  [Zone index]:  20
DATACENTER_TOP_ZN_6  [Zone index]:  21
TOPFLOOR_PLENUM  [Zone index]:  22
################################################
```

Figure 8: Importing OfficeLarge in `BuildingEnv`.

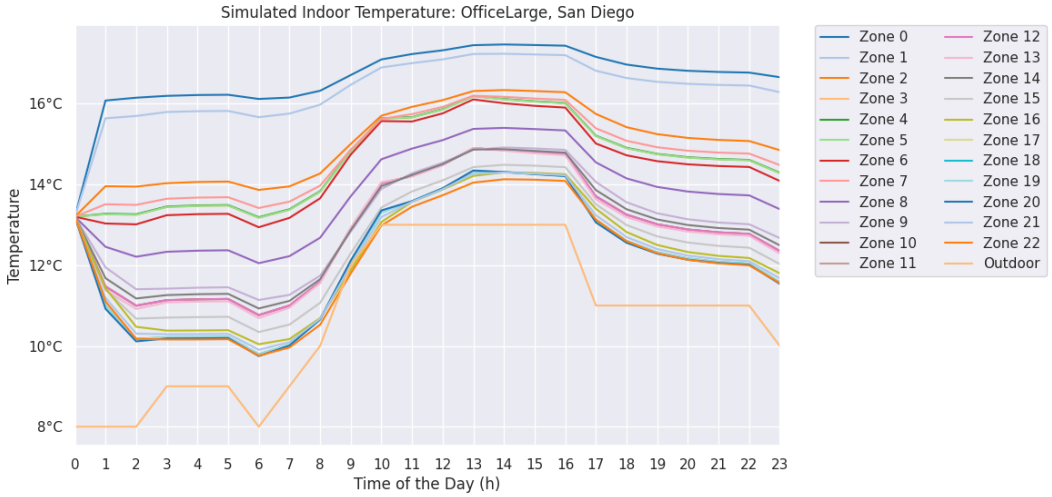


Figure 9: Simulated indoor temperature in different zones of OfficeLarge for 1 day without control.

lookahead window w, the MPC controller solves the following optimization problem,

$$\min_{a(t),\ldots,a(t+w-1)} \quad \sum_{k=t}^{t+w-1} (1-\beta)\left\|a(k)\right\|_2 + \beta\|T^{\text{target}}(k) - T(k)\|_2 \tag{8a}$$

$$\text{s.t.} \quad -1 \preceq a_i(k) \preceq 1\,, \forall i \in \{0,\ldots,M\}, \forall k \in \{t,\ldots,t+w-1\} \tag{8b}$$

$$\text{building dynamics in (7)}\,, \tag{8c}$$

and implements action $a^\star(t)$. MPC method has complete model knowledge and perfect predictions about building occupancy $[N_1, N_2, \ldots, N_M]$, the ground and environmental temperature $T_{\text{G}}(k), T_{\text{E}}(k)$, solar irradiance $Q^{\text{GHI}}(k)$ for $k = t, t+1, \ldots, t+w-1$. (8b) ensures that the control signals are feasible, and (8c) follows the building physics model in (7) which uses a zero-order hold discretization method to derive from the continuous-time model with discretization interval as $\Delta T = 1/12$ hour.

Multiagent Environment. The multiagent setting pairs each zone with one agent, so that each agent i is only responsible for action $a_i(t)$. Currently, we let every agent observe the complete building state space, and each agent receives the same global reward. For future directions, we plan

on implementing a separate local state space for each agent and also imposing a total power constraint (*e.g.*, $\|a(t)\|_1$ must be bounded by some limit).

Training We trained Proximal Policy Optimization (PPO), Soft Actor-Critic (SAC), and Advantage Actor Critic (A2C) [40] Reinforcement Learning algorithms on the `BuildingEnv` environment using PyTorch and StableBaselines3. These algorithms were applied to an "OfficeSmall" type building situated in a "Hot Dry" climate, specifically in Tucson. The training and testing periods were set in the winter (January 2003) and summer (June 2004) sections, respectively. The models were trained during both the winter and summer sections, but testing was conducted solely in the winter section to assess the impact of distribution shifts. We experimented with two learning rates for each of PPO, SAC, and A2C: 3e-4 and 3e-5 for PPO and SAC, and 7e-4 and 3e-4 for A2C. The training used a time resolution of 300 seconds (or 5 minutes). The model demonstrating the best performance in training was subsequently selected for further analysis.

Representativeness and Generalizability `BuildingEnv` is designed to ensure broad generalizability across diverse building scenarios. We have integrated an extensive collection of standard prototype building models coupled with a variety of weather conditions, encompassing the majority of Reference Building types. These models are based on the Department of Energy (DOE)'s Commercial Reference Building Models, offering a robust foundation for simulation. Furthermore, the environment is highly customizable. Users have the flexibility to adjust parameters such as the wall material, the solar heat gain coefficient of windows, and the ground temperature specific to the building's location. For researchers and developers seeking even more specificity, there is also an option to define custom building models using EnergyPlus. Users can then input the resulting output input data file (IDF), along with the corresponding EPW weather file, into `BuildingEnv`, providing a custom simulation experience.

Limitations Beyond the modeled weather variations, occupancy dynamics also introduce distribution shifts in building energy management. While `BuildingEnv` provides mechanisms to simulate occupants in different zones with set activity schedules, truly replicating the unpredictability of human behavior remains a complex endeavor. For instance, we can configure a room to have four individuals engaged in seated work from 1pm to 4pm, followed by three individuals running from 4pm to 5pm, with the room being vacant after 5pm. Despite these features, there are certain distribution shifts that `BuildingEnv` does not currently address, such as equipment failures, external environmental impacts like nearby construction or urban heat island effects, and the nuances of building aging. However, it is worth noting that the modular design of `BuildingEnv` provides a foundation that is conducive to future enhancements and adaptations, allowing for the incorporation of additional sources of distribution shift as our understanding of building dynamics evolves.