Deploying Reinforcement Learning based Economizer Optimization at Scale

Jiarong Cui, Wei Yih Yap, Charles Prosper, Bharathan Balaji, Jake Chen
{ivancui,yihyap,cpro,bhabalaj,jakechen}@amazon.com
Amazon

ABSTRACT

Building operations account for a significant portion of global emissions, contributing approximately 28% of global greenhouse gas emissions, according to the International Energy Agency. With the anticipated increase in cooling demand due to rising global temperatures, the optimization of rooftop units (RTUs) in buildings becomes crucial for reducing energy consumption and associated emissions. We focus on the optimization of the economizer logic within RTUs, which balances the mix of indoor and outdoor air. By effectively utilizing free outside air when available, RTUs can significantly decrease mechanical energy usage, leading to reduced energy costs and emissions. However, the current practice of economizer optimization relies on static guidelines set by ASHRAE, which overlook the specific conditions and dynamics of individual facilities. We introduce a reinforcement learning (RL) approach that adaptively controls the economizer based on the unique characteristics of individual facilities. We have trained and deployed our solution *in the real-world* across a geographically distributed building stock. We address the scaling challenges with our cloud-based RL deployment on 10K+ RTUs across 200+ sites.

KEYWORDS

hvac, reinforcement learning, optimization, economizer

ACM Reference Format:
Jiarong Cui, Wei Yih Yap, Charles Prosper, Bharathan Balaji, Jake Chen. 2023. Deploying Reinforcement Learning based Economizer Optimization at Scale. In *Proceedings of Fourth ACM SIGEnergy Workshop on Reinforcement Learning for Energy Management in Buildings Cities (RLEM '23).* ACM, New York, NY, USA, 4 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

As global temperatures continue to rise, the demand for cooling in buildings is expected to increase, especially in regions with hot climates [7]. Without energy-efficiency measures, higher cooling demand will result in even higher emissions. We focus on economizer optimization in rooftop units (RTUs), typically deployed in industrial HVAC systems. We experiment on a site with 10K+ RTUs located in a hot region. The economizer balances the indoor and outdoor air during the conditioning process. By leveraging free cold air, the economizer can significantly reduce mechanical energy usage, leading to decreased energy costs and associated emissions.

The standard practice for economizer optimization follows the guidelines set by the American Society of Heating, Refrigerating and Air-Conditioning Engineers (ASHRAE). These guidelines rely on hard-coded setpoints based on regional climate data [5]. This standardized does not account for the specific conditions and dynamics of individual facilities, and therefore, overlooks valuable opportunities to capitalize on free cooling based on real-time conditions. To overcome these limitations and propel energy optimization in facilities, recent advancements in reinforcement learning (RL) have emerged as promising solutions [4, 9]. We present an RL approach to create self-adapting economizer setpoints, that adjust to the unique characteristics of individual facilities in real-time. Additionally, these RL models are regularly updated, incorporating new data and insights to continuously enhance their performance.

Contributions: Prior works primarily focus on zonal setpoint changes (e.g. [8]), to our knowledge we are the first to focus on economizer optimization. Further, prior works demonstrate RL solutions in simulation [9] or a single building [8], we deploy our solution to 200+ sites with 10K+ RTUs. To our knowledge we are the first to discuss cloud-based deployment of RL based control at scale. We discuss solutions to challenges across four axes in our deployment: data standardization, storage scalability, compute scalability, and human-in-the-loop control.

We open-source our code with a permissive license[1].

2 REINFORCEMENT LEARNING

In RL, an agent interacts with an environment described by a Markov Decision Process (MDP). The agent selects actions based on the current state of the environment and receives feedback in the form of the updated state and associated rewards. RL has been extensively explored for building optimization tasks [4, 9].

We use the well-established Soft Actor-Critic (SAC) algorithm [6], which combines policy-based (Actor) and value-based (Critic) RL methods. SAC utilizes the state-value function to estimate long-term accumulated reward expected from a given state, and an action-value function to estimate cumulative rewards for state-action pairs. Additionally, SAC employs an advantage function to measure the advantage of an action compared to the average expected value at a state. The Actor's policy neural network is updated using policy gradients and maximizes rewards, while the critic neural network is refined to better approximate the true value function. Notably, SAC introduces an entropy regularization term to encourage exploration and strike a balance between expected rewards and entropy. This inclusion makes SAC an appropriate choice as the algorithm

RLEM '23, November 15 – 16, 2023, Istanbul, Turkiye
© 2023 Association for Computing Machinery.
ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/XXXXXXX.XXXXXXX

[1]https://github.com/aws-solutions-library-samples/guidance-for-monitoring-and-optimizing-energy-usage-on-aws

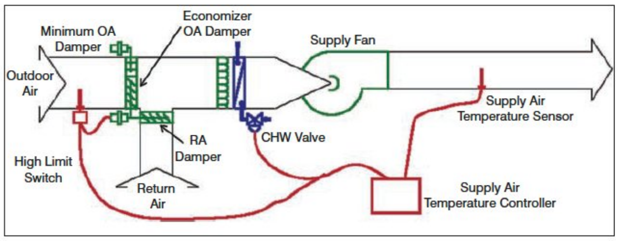

Figure 1: Illustration of an RTU equipped with an economizer

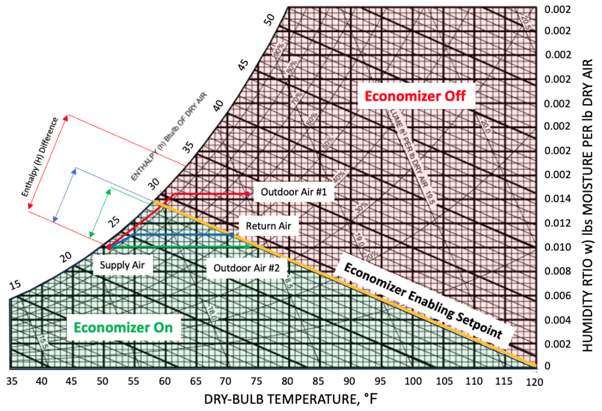

Figure 2: Psychrometric Chart for Economizer Operation

explores and learns efficiently in environments with stochasticity. Our experiments with SAC showed consistent improvement over the existing rule-based control, and we deemed it sufficient for deployment without further experimentation due to development costs. However, other RL algorithms can be incorporated into our framework as a drop-in replacement.

3 PROBLEM FORMULATION

Figure 1 shows an overview of economizer operation. The economizer reduces the load of cooling on the system, and we model how much energy can be saved with free cooling using available sensor measurements. Our economizers are equipped to monitor outdoor temperature and humidity, and adjusts the damper position based on a setpoint. Damper position is maintained at a minimum of 10% to ensure sufficient air exchange. Economizers effectively reduce the cooling load by minimizing the enthalpy required to reach the desired supply air temperature setpoint.

Figure 2 shows the psychrometric chart for economizer operation, which captures the relationships between temperature, humidity, and enthalpy. The neural network in our RL agent learns these relationships, and optimizes the economizer for the local weather patterns by maximizing the long-term reward.

We formulate the following MDP in OpenAI gym [3]:

State: Outdoor temperature, outdoor humidity, return temperature and return humidity. These states include the features needed for air ratio calculation (%OA) and enthalpy (H) interpolation using psychrometric charts. We assume supply air temperature and humidity to be constant. Table 1 lists all the constants in our environment.

Table 1: Constants in the RL environment.

Name of Variable	Value (Unit)
Supply Temperature (T_{supply})	55 (°F)
Supply Humidity ($\mathcal{H}_{supply}$)	0.5 (PPM)
Minimal Economizer Enthalpy Setpoint ($H_{min-econ}$)	20 (BTU/lb)
Maximal Economizer Enthalpy Setpoint ($H_{max-econ}$)	30 (BTU/lb)
Min. Economizer Temperature Setpoint ($T_{min-econ}$)	40 (°F)
Max. Economizer Temperature Setpoint ($T_{max-econ}$)	75 (°F)
Minimal Outside Air Ratio (OA_{min})	0.1
Supply Airflow (U)	8000 (CFM)
Cooling Enable Setpoint ($Cool_{sp}$)	55 (°F)

Action: The action space consists of 2 setpoints: economizer maximum temperature ($T_{max-econ}$) and economizer maximum enthalpy ($H_{max-econ}$), both in continuous space. We constrained the action space between [20, 30] and [40, 75] for the 2 setpoints respectively. We select actions every hour.

Reward: Our reward function estimates the power required to meet the required supply air temperature with the given economizer status using the Total Heat Gain equation with minor modifications, and penalize high difference of entering and leaving air enthalpy, which indicates longer run time for mechanical cooling. At any given time step t, our reward function is:

$$reward = (H_{mixed} - H_{supply}) * airflow * 4.5/100000 \tag{1}$$

where, H_{mixed} is the mixed air enthalpy, H_{supply} is the supply air enthalpy, $airflow$ is measured in CFM, and the constants account for the standard air density, and the conversion factor from minutes to hours. The episode length is set to 200 steps so that the RL agent maximizes the long-term discounted rewards. The reward accumulation occurs as part of the SAC critic loss function.

We calculate the enthalpy and estimate the percentage of outside air (%OA) in the mixed air, and make control decision with actions selected by the SAC model. The enthalpy of the mixed air can be estimated using temperature and humidity measurements. We use the following equation:

$$H = 0.240 * T + \mathcal{H} * ((0.444) * T + 1061) \tag{2}$$

where H is enthalpy in Btu/lb, $\mathcal{H}$ is humidity and T is temperature in °F. We use the same equation to compute H_{supply}, H_{return} and H_{mixed}, with corresponding values of temperature and humidity.

The economizer system utilizes sensors to monitor the return temperature (T_{return}) and return air humidity ($\mathcal{H}_{return}$). Additionally, outdoor temperature ($T_{outdoor}$) and humidity ($\mathcal{H}_{outdoor}$) sensors provide information about the external conditions. When the outside air enters through the damper, it mixes with the return air, resulting in a mixed air stream that passes through the heating and cooling coil before being supplied to the facility. The temperature of this mixed air is denoted as T_{mixed}. Since most RTUs lack sensors for measuring mixed air temperature and humidity directly, an approximation can be made using the formula below:

$$\%OA = (T_{return} - T_{supply})/(T_{return} - T_{outdoor}) \tag{3}$$

The $\%OA$ is used to compute T_{mixed} and $\mathcal{H}_{mixed}$ as shown in Algorithm 1.

In the case of differential enthalpy control with fixed dry-bulb temperature, the economizer enabling setpoints depend on the maximum enthalpy and maximum dry-bulb temperature allowed for the economizer. This results in the division of the psychrometric chart into two distinct operating regions as shown in Figure 2.

In our formulation, the hourly time step and the energy-based reward function does not necessitate the use of an RL solution because each time step can be optimized independently. However, the use of neural networks with deep RL does make even this greedy optimization much easier to solve compared to rule-based control as the agent learns the non-linearity shown in the psychrometric chart and automatically optimizes control for a specific site. The MDP framework also makes it easy to switch to a dynamic pricing or carbon intensity based reward in a future deployment, which does have a dependency across time steps.

4 VALIDATION WITH HISTORICAL DATA

We collected historical outdoor temperature, outdoor humidity, return temperature, and return humidity data from an RTU, recorded every 15 minutes from May 2018 to October 2021. We filtered out records that were either missing or anomalous, converted both outdoor and return humidities to decimal value and clipped them between range of [1e-5, 1], and resampled the data into hourly granularity by taking the average. We hold out 1 month of data for validation. After data cleaning and preprocessing, the processed dataset contains around 12K records. We trained our SAC agent for a total of 200K time steps. Algorithm 1 describes each step in the process, with f denoting Equation 2.

Algorithm 1:

```
Input  : $T_{outdoor}, \mathcal{H}_{outdoor}, T_{return}, \mathcal{H}_{return}, max\_steps$
Output : R, action, SAC
Initialize: SAC, steps, RL agent SAC
for each step do
    $state = T_{outdoor}, T_{return}, \mathcal{H}_{outdoor}, \mathcal{H}_{return}$
    $action = SAC(state)$
    $H_{supply} = f(T_{supply}, \mathcal{H}_{supply})$
    $H_{outdoor} = f(T_{outdoor}, \mathcal{H}_{outdoor})$
    $H_{return} = f(T_{return}, \mathcal{H}_{return})$
    if $(H_{outdoor} > H_{max-econ})$ or $(T_{outdoor} > T_{max-econ})$ then
        $\%OA = OA_{min}$
    else
        $\%OA = (T_{return} - T_{supply})/(T_{return} - T_{outdoor})$
    $H_{mixed} = \%OA * H_{outdoor} + (1 - \%OA) * H_{return}$
    $T_{mixed} = \%OA * T_{outdoor} + (1 - \%OA) * T_{return}$
    $R = ((H_{mixed} - H_{supply}) * U * 4.5))/100000$
    $SAC = update(SAC, state, action, R)$
    $steps = steps + 1$
    if $(steps > max\_steps)$ then
        break
```

Figure 3 shows the power consumption between different strategies in updating economizer setpoints for one RTU. On the y-axis is the average power consumption over the validation period, so lower the better. We compare against two baselines: no economizer, and the ASHRAE standard. The ASHRAE standard uses rules based on climate zones to approximately pick the appropriate setpoint for

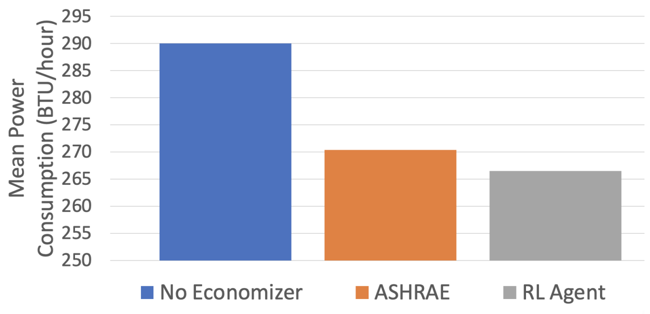

Figure 3: Performance comparison of our RL agent against ASHRAE standard policy for one RTU.

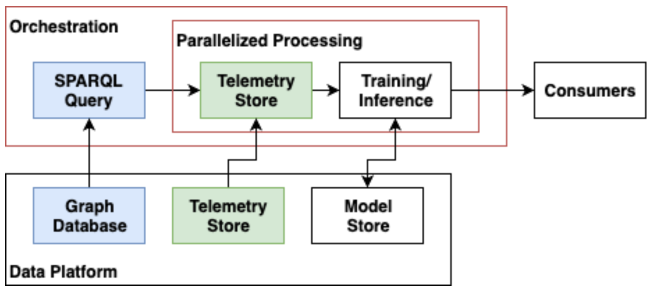

Figure 4: Cloud deployment architecture

an economizer based on the psychrometric charts. The RL agent outperforms the ASHRAE standard by resolving the non-linearities in psychrometry and selecting a more precise setpoint that maximizes energy savings. Overall, the RL agent yeilds 5% reduction in average power use.

We deployed the solution on 10K+ RTUs across 200+ sites for hourly inferencing. We train one agent per site, but use the same policy for all the RTUs in a site. We re-train the RL agent each quarter to account for drift in environment conditions. We redact online deployment results due to proprietary reasons.

5 DEPLOYMENT ARCHITECTURE

We leverage cloud services to scale our deployment, Figure 4 shows our architecture. We discuss the challenges in scaling and the solutions employed below.

5.1 Data standardization

Building data is often disorganized, with inconsistent naming conventions for sites, assets, and sensors due to various factors such as different teams, vendors, or engineers involved during setup. As a result, deploying downstream analytics, including RL policies, becomes a time-consuming process that requires customization per each site, and thus limits scalability. These challenges are well-known, and have been documented in prior works [1, 2].

To address the data standardization challenge, we leveraged the Brick schema, which provides a well-defined ontology for building-related data, ensuring consistency and uniformity in data representation and structure [1]. We load standardized sites, RTUs, sensors, and their relevant relationships into a Graph Database. We use

SPARQL queries to retrieve relevant sites, assets, and points associated with our MDP. A sample query is shown below.

```
SELECT ?point ?timerseries_id
WHERE {
  ?site BRICK:hasName "site1" .
  ?rtu BRICK:hasName "RTU1" .
  ?site BRICK:hasLocation ?rtu .
  ?rtu a BRICK:RTU .
  ?rtu BRICK:hasPoint ?point .
  ?point a BRICK:Zone_Air_Temperature_Setpoint .
  ?point BRICK:timeseries ?timerseries_id .}
```

5.2 Data scalability

Traditional relational databases are not designed to simultaneously handle both the long-term requirements for model training alongside real-time requirements for model inference and analytics, particularly in our case of multiple years of minute to sub-minute level data from millions of sensors. While certain data architecture optimizations like redundancy, joins, partitioning, and other techniques may be effective in specific use cases, the ever-changing and adhoc nature of analytics continually introduces new requirements.

We utilize a combination of long-term (cold) storage and short-term (hot) storage to ensure a robust and scalable data management approach for model training, model inference, and other ad-hoc analytical use cases. For long-term storage of training data that is only accessed for quarterly retraining, we leverage Block Storage to store years of data at the lowest granularity possible. This method is cost efficient for infrequent reads yet allows for the transformation of the raw data into other storage systems for analytical use cases outside of energy optimization. For model inference and real-time analytics, we store incoming streaming sensor data into purpose-built timeseries SQL databases. These tools are specifically designed to handle the simultaneous reading and writing of high-volume and high-frequency sensor data.

After retrieving the relevant data points required from a SPARQL query (results include a timeseries id), we parallelize the SQL queries to retrieve the time series data across sites and RTUs efficiently. Each query focuses on a specific site and RTU combination. By combining the power of a Graph Database and parallel SQL queries to timeseries databases, we effectively leverage standardized data to scale our deployment.

5.3 Compute scalability

We aim to concurrently perform inference across hundreds of sites every hour, and run model training every quarter. We rely on the serverless capabilities provided by Cloud compute to meet these scaling requirements.

In our use case, the SPARQL queries would return a list of sites with a list of RTUs per site. During model training, we use serverless orchestration tools to map training jobs to every RTU associated with a site, resulting in tens-of-thousands of concurrent jobs. We process the concurrent training jobs using a batch processing service, which is suited for GPU intensive process of environment simulation and subsequent training of the RL agent. We store the resulting policies in long-term storage for later model inference. These concurrent training jobs take up to an hour each.

During model inference, we also use workflow orchestration to map training jobs to every RTU associated with a site. However, we process the concurrent jobs using function-as-a-service (FaaS) instead of batch processing. We use CPUs in FaaS to reduce costs and availability issues, and complete concurrent inferences within 5 minutes on average.

Using this architecture, we have already scaled to 10K+ RTUs, and do not foresee difficulties in continuing to scale. The cost of our deployment is up to $50K/year, and we estimate saving >$250K in annual energy bills.

5.4 Human-in-the-loop Control

Although we have been able to automatically recommend economizer setpoints across hundreds of sites, we have not yet automated the command and control of the RTUs themselves. Instead, we output the recommended setpoints to a web-based front-end that displays the setpoints as well as the potential energy savings. Building operators review and input these recommended setpoints manually. The reasons for this are twofold. First is the ongoing security discussions associated with automating command-and-controls. While integration with the command and control suite will allow setpoint automation, it would also grant access to more critical setpoints that may disable the RTU completely. The second reason is the novelty and subsequent lack of trust with the algorithm, especially given the complexity of an RL solution. We are currently evaluating the performance with end users and stakeholders to establish trust in the recommended setpoints. Once the security issues are resolved, we expect building operators will slowly taper off the manual validations and allow more setpoints to be automated, at which time we can expect to fully realize the energy savings demonstrated with simulation.

REFERENCES

[1] Bharathan Balaji, Arka Bhattacharya, Gabriel Fierro, Jingkun Gao, Joshua Gluck, Dezhi Hong, Aslak Johansen, Jason Koh, Joern Ploennigs, Yuvraj Agarwal, et al. 2016. Brick: Towards a unified metadata schema for buildings. In *Proceedings of the 3rd ACM International Conference on Systems for Energy-Efficient Built Environments*. 41–50.

[2] Bharathan Balaji, Chetan Verma, Balakrishnan Narayanaswamy, and Yuvraj Agarwal. 2015. Zodiac: Organizing large deployment of sensors to create reusable applications for buildings. In *Proceedings of the 2nd ACM international conference on embedded systems for energy-efficient built environments*. 13–22.

[3] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. 2016. Openai gym. *arXiv preprint arXiv:1606.01540* (2016).

[4] Can Cui, Chunxiao Li, and Ming Li. 2022. An Online Reinforcement Learning Method for Multi-Zone Ventilation Control With Pre-Training. *IEEE Transactions on Industrial Electronics* 70, 7 (2022), 7163–7172.

[5] SI Edition, DH Erbe, MD Lane, SI Anderson, PA Baselici, S Hanson, R Heinisch, J Humble, S Taylor, and R Kurtz. 2010. Energy standard for buildings except low-rise residential buildings. *ASHRAE, Atlanta, GA, USA, Tech. Rep. IP Edition* (2010).

[6] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. 2018. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*. PMLR, 1861–1870.

[7] Ammar M Khourchid, Salah Basem Ajjur, and Sami G Al-Ghamdi. 2022. Building cooling requirements under climate change scenarios: Impact, mitigation strategies, and future directions. *Buildings* 12, 10 (2022), 1519.

[8] Hsin-Yu Liu, Bharathan Balaji, Sicun Gao, Rajesh Gupta, and Dezhi Hong. 2022. Safe hvac control via batch reinforcement learning. In *2022 ACM/IEEE 13th International Conference on Cyber-Physical Systems (ICCPS)*. IEEE, 181–192.

[9] Liang Yu, Di Xie, Chongxin Huang, Tao Jiang, and Yulong Zou. 2018. Energy optimization of HVAC systems in commercial buildings considering indoor air quality management. *IEEE Transactions on Smart Grid* 10, 5 (2018), 5103–5113.