UNIVERSITY OF CALIFORNIA SAN DIEGO

# Policy Regularization in Model-Free Building Control via Comprehensive Approaches from Offline to Online Reinforcement Learning

A dissertation submitted in partial satisfaction of the requirements for the degree
Doctor of Philosophy

in

Electrical Engineering (Computer Engineering)

by

Hsin-Yu Liu

Committee in charge:

Professor Rajesh K. Gupta, Chair
Professor Sicun Gao
Professor Pat Pannuto
Professor Yuanyuan Shi

2024

The dissertation of Hsin-Yu Liu is approved, and it is acceptable in quality and form for publication on microfilm and electronically.

University of California San Diego

2024

# DEDICATION

Dedicated to my new family for being with me through these difficult days. And thank you to my previous family for giving me the strength to face it all.

# EPIGRAPH

*Wer ein Warum zu leben hat, erträgt fast jedes Wie.*

—Friedrich Nietzsche

## TABLE OF CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

## ACKNOWLEDGEMENTS

I wish to express my profound gratitude to Professor Rajesh Gupta for his steadfast support and unwavering trust throughout my academic journey. His gracious acceptance of me as a transferred student, coupled with the invaluable freedom he bestowed upon me to explore my research interests, has played a pivotal role in my academic development. Professor Gupta’s provision of resources and opportunities for collaboration has not only facilitated the advancement of my research but also enabled its recognition and promotion within the academic domain. Under his exemplary leadership, I have found a nurturing environment to learn, grow, and thrive, for which I am deeply thankful.

I would like to express my special gratitude to Dr. Bharathan Balaji for his significant contributions to my doctoral research. He displays remarkable dedication in addition to his extraordinary academic achievements. He would review every submission, scrutinizing every word and providing suggestions of great insight with unwavering commitment. From his mentorship, I learned how to deeply analyze research questions. He was not only my mentor but also my friend, offering considerable personal support throughout my journey. It was a privilege to work with him and I am honored to work with such a researcher in my career.

My sincere appreciation to Dr. Dezhi Hong for building up my knowledge in the building domain, the techniques for the design of experiments, and a detail-oriented work style. This leads me to the domain of building research.

I extend my heartfelt gratitude to Professor Sicun Gao, whose course on ’*Search and*

*Optimization*' not only ignited my passion for Reinforcement Learning but also steered my academic journey toward Batch Reinforcement Learning, laying the groundwork for my Ph.D. research.

Thanks to my Ph.D. committee members: Professor Sicun Gao, Pat Pannuto, and Yuanyuan Shi. They provided me with valuable input and feedback on my research work throughout my Ph.D. study.

To all my labmates and alumni: Xiaohan Fu for the support of BuildingDepot APIs, Brick server, and all the data of our on-campus buildings. Xiyuan Zhang, Ranak Roy Chowdhury, Shuheng Li, and Jiayun Zhang for all the smiles and memories, I have learned a lot from your research. Also, Dr. Jason Beomkyu Koh and Dr. Dhiman Sengupta for their guidance. Dr. Jeng-Hau Lin for all the advice.

To my cherished new family, whose unconditional love and support have been the quiet yet formidable pillars through the countless hours of this Ph.D. journey: To my beloved wife, Szu-Chieh Chen (Sylvia), whose patience and strength have been the sanctuary of my aspirations. You have been my steadfast companion, nurturing our home with warmth and understanding, even when academic pursuits demanded my presence elsewhere. To my daughter, Tsai-Fei Liu (Phoenix), whose laughter and wonder have been a constant reminder of the joy and curiosity that underpin the pursuit of knowledge. Your innocence and vitality have been the light that guided me through moments of doubt and weariness. And to our loyal companion, Bella, a golden female Pomeranian mixed with Chihuahua, whose silent companionship and comforting presence offered solace during the long nights of study and writing.

To my Father, Han-Sheng Liu (Hans), who has always served as a moral compass for me; to my Mother, Jen-Tai Tang (Tracy), who sadly did not have the chance to witness the culmination of my Ph.D. journey; to my sister, Hsin-Chieh Liu (Marcy), and to my brother, Hsin-Kai Liu (Kevin), who were integral to my childhood—growing up alongside me, creating cherished memories, and imparting valuable knowledge.

Together, you have all sacrificed in innumerable ways, often without recognition. It is to you I owe the fruition of my dreams and the completion of this chapter. This journey has been as much yours as it has been mine, and it is with a heart full of love and gratitude that I dedicate this work to you. May this dissertation stand as a testament to the power of love and sacrifice, and may it honor the silent yet profound contributions you have made.

Chapter 2, in part, is a reprint of the material that appears in the proceeding of ACM/IEEE 13th International Conference on Cyber-Physical Systems (ICCPS 2022). By authors Hsin-Yu Liu, Bharathan Balaji, Sicun Gao, Rajesh Gupta, and Dezhi Hong with the title - ”Safe HVAC control via batch reinforcement learning”. The dissertation author is the primary investigator and author of this paper.

Chapter 3, in part, is a reprint of the material that appears in the proceeding of the 9th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation (BuildSys’ 2022). Bu authors Hsin-Yu Liu, Xiaohan Fu, Bharathan Balaji, Rajesh Gupta, and Dezhi Hong with the title - “B2RL: an open-source dataset for building batch reinforcement learning.” The dissertation author is the primary investigator and author of this paper.

Chapter 4, in part, is a reprint of the material that appears in the proceeding of the 14th ACM International Conference on Future Energy Systems (e-Energy 2023). By authors Hsin-Yu Liu, Bharathan Balaji, Rajesh Gupta, and Dezhi Hong with the title - “Rule-based Policy Regularization for Reinforcement Learning-based Building Control” The dissertation author is the primary investigator and author of this paper.

Chapter 5, in part, is a reprint of the material that will be submitted in the future by the authors Hsin-Yu Liu, Bharathan Balaji, Rajesh Gupta, and Dezhi Hong. with a tentative title: “Policy Regular- ization for Offline-to-Online Reinforcement Learning in HVAC Control”. The dissertation author is the primary investigator and author of this paper.

Appendix A.1, in part, is a reprint of the material that appears in the Offline Reinforcement Learning Workshop at Neural Information Processing Systems (NeurIPS Offline-RL Work- shop 2021), vol. 2021, 2021. By author Hsin-Yu Liu, Bharathan Balaji, Rajesh Gupta, and Dezhi Hong with the title - “Offline reinforcement learning with Munchausen regularization.” The dissertation author is the primary investigator and author of this paper.

# VITA

| | |
|---|---|
| 2001-2005 | Bachelor of Science in Physics, National Central University, Taiwan |
| 2006-2008 | Master of Science in Optoelectronics, National Taiwan University, Taiwan |
| 2018-2024 | Doctor of Philosophy in Electrical Engineering (Computer Engineering), University of California San Diego |

# PUBLICATIONS

Hsin-Yu Liu, Bharathan Balaji, Rajesh Gupta, and Dezhi Hong. “Policy Regularization for Offline-to-Online Reinforcement Learning in HVAC Control”, (Future submission).

Hsin-Yu Liu, Bharathan Balaji, Rajesh Gupta, and Dezhi Hong. “Rule-based Policy Regularization for Reinforcement Learning-based Building Control”, *14th ACM International Conference on Future Energy Systems* (e-Energy 2023), pp. 242-265. 2023.

Hsin-Yu Liu, Xiaohan Fu, Bharathan Balaji, Rajesh Gupta, and Dezhi Hong. “B2RL: an open-source dataset for building batch reinforcement learning.”, *9th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation* (BuildSys’ 2022), pp. 462-465. 2022.

Hsin-Yu Liu, Bharathan Balaji, Sicun Gao, Rajesh Gupta, and Dezhi Hong. “Safe HVAC control via batch reinforcement learning.”, *ACM/IEEE 13th International Conference on Cyber-Physical Systems* (ICCPS 2022), pp. 181-192. IEEE, 2022.

Hsin-Yu Liu, Bharathan Balaji, Rajesh Gupta, and Dezhi Hong. “Offline reinforcement learning with Munchausen regularization.”, *Offline Reinforcement Learning Workshop at Neural Information Processing Systems* (NeurIPS Offline-RL Workshop 2021), vol. 2021, 2021.

ABSTRACT OF THE DISSERTATION

**Policy Regularization in Model-Free Building Control**
**via Comprehensive Approaches**
**from Offline to Online Reinforcement Learning**

by

Hsin-Yu Liu

Doctor of Philosophy in Electrical Engineering (Computer Engineering)

University of California San Diego, 2024

Professor Rajesh K. Gupta, Chair

Reinforcement Learning (RL) has been extensively explored in building control, primarily because the problems in this field can be effectively formulated as Markov Decision Process (MDP) problems. Traditional approaches predominantly treat these challenges as online RL problems, assuming that accurate simulators or environmental models are already established and fine-tuned. However, creating and calibrating these models is not only time-intensive and resource-heavy but also starting from a randomly initialized policy

could pose safety concerns.

Consequently, to address real-world issues, data-driven strategies are more practical alternatives for learning agents. This is particularly relevant in contemporary building management systems, where control and actuation data are systematically archived. Such data can serve as a valuable foundation for prior knowledge and be stored as experience replays, enabling agents to learn and adapt more effectively. Typically, a default building control policy is crafted by domain experts leveraging their best-known practices. This expert policy can serve as the expert demonstration, providing a behavioral guide that informs and enhances the early performance of a learning agent, thereby minimizing opportunity costs.

Nevertheless, the policy learning by offline methods is limited due to the static dataset the agent learns from. No further exploration in the state-action spaces is possible. Thus, it is crucial to study the offline-to-online methods to improve the pre-trained offline models with online interaction. The major challenge of offline-to-online methods is to overcome the extrapolation errors in value estimation encountered during the distribution drift from the static experience replay to the environments to be evaluated.

In this dissertation, we introduce studies encompassing a suite of data-driven approaches in building control, beginning with offline/batch reinforcement learning. Where we adapt the Kullback-Leibler divergence to penalize the policy updates that deviate far from their previous selves. Also, we build and make available the first open-source building control dataset for batch reinforcement learning benchmark. A standardized dataset is crucial for batch reinforcement learning, Then, we explore a unified policy regulariza-

tion method that integrates existing policies within both online and offline frameworks. It provides robustness and stability to reinforcement learning. Finally, we extend our exploration into offline-to-online reinforcement learning and address the challenge of adapting the distribution drift with adaptive policy regularization to automatically tune the agent learning. Collectively, this dissertation studies the policy regularization in model-free building control with comprehensive approaches from offline to online reinforcement learning. With important conditions that will guide the design and operation of cyber-coupled systems driven by sensory data.

# Chapter 1

# Introduction

Buildings account for 30% of energy use worldwide, and approximately half of it is ascribed to HVAC systems [115]. Control systems are critical to managing these. Reinforcement Learning (RL) has improved upon traditional control methods in increasing the energy efficiency of HVAC systems. Most earlier works in the area are on RL learning in an online paradigm [87, 126, 24, 38, 139, 146]. These online methods require configuring complex thermal simulators to train the RL models by interacting with the learning agent with the simulator during the training and evaluation before real-world deployment. Simulators such as EnergyPlus [18] and TRNSYS [53] are used to simulate the thermal states of a building. Designing and calibrating such models for a large building is time-consuming and requires expertise. Furthermore, it is inefficient to transfer these models to other buildings due to the differences in geometry, weather, materials, thermal dynamics of the HVAC systems, and so forth. The alternative way is to use historical data-driven thermal models. Even though it can take more than ten thousand time steps to reach the performance of the rule-based control (RBC) policy [24].

In real-world scenarios, most large buildings are controlled via building management systems (BMS), where thermal data can be stored in a database. With advances in sensing technologies and machine learning, data-driven models have been more popular in recent research. Batch reinforcement learning (BRL) or offline reinforcement learning, a data-driven approach that learns from static datasets generated with unknown behavioral policy, has not been explored widely in the building control community. BRL models are capable of learning a better policy than the behavioral policy without accurate environment models or simulation environments as oracles.

Nevertheless, due to the limited exploration, when the agent further explores the uncharted state-action spaces, it might lead to inaccurate value estimation due to the lack of support.

## 1.1 Reinforcement Learning

Reinforcement Learning problems can be formulated as a Markov Decision Process (MDP), a sequential decision-making problem that aims to maximize the discounted accumulative rewards. The MDP consists of a tuple: $\mathcal{M} = (\mathcal{S}, \mathcal{A}, p, r, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $p$ is the transition dynamics. The next state $S_{t+1} \sim p(\cdot|S_t, A_t)$, is decided by the current state and the action selected by a policy $\pi(a|s), \pi : \mathcal{S} \to \mathcal{A}$ either in a stochastic or a deterministic fashion. The reward function $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}, r \in \mathbb{R}$ is mapped as a scalar, and the discount factor $\gamma \in [0, 1)$. The agent's goal is to optimize the policy to maximize the discounted accumulated return $\mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t r_t]$ [109].

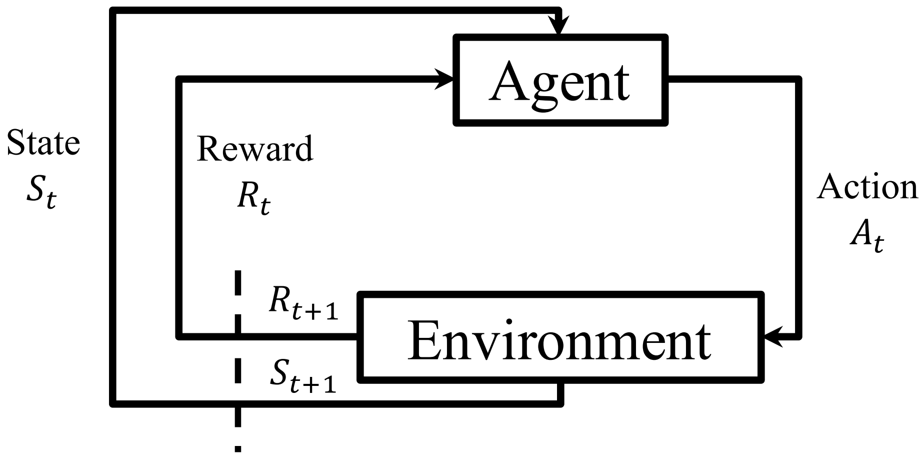


**Figure 1.1**: The agent-environment interaction in a Markov decision process.

## 1.2 Building Control

Building control systems represent a critical juncture in the evolution of urban infrastructure, embodying the confluence of technological advancement and sustainable development. In the era of smart cities and green buildings, the optimization of heating, ventilation, and air conditioning (HVAC) systems transcends operational efficiency, it also embodies a commitment to environmental stewardship, energy conservation, and the promotion of health and well-being within indoor spaces. The significance of these systems is underscored by their substantial energy footprint; HVAC systems account for approximately 40% of energy consumption in commercial buildings [83], making them prime targets for efficiency improvements.

Recent years have witnessed a paradigm shift in building control strategies, propelled by the integration of Internet of Things (IoT) technologies and the advent of sophisticated Machine Learning algorithms. This fusion of digital and physical domains offers unprecedented opportunities to enhance building automation systems, enabling them to learn from and adapt to changing environmental conditions and occupant behaviors in real-time [147]. The potential of such intelligent systems to reduce energy consumption while improving occupant comfort and health is immense, marking a significant step forward in the quest for more sustainable and livable urban environments.

## 1.3 Organization

This dissertation is organized as follows: In Chapter 2, we present our work for deploying the BRL agents in the real-world building HVAC control systems for the first time in the domain. The details of the algorithm development are described in Appendix A.1, where the algorithms are tested with the OpenAI [13] MuJoCo robotic control tasks.

Chapter 3, encloses the first open-source BRL dataset for building HVAC control - *B2RL*, which includes one dataset extracted from the real-world sensing and actuation data in the University of California - San Diego Computer Science and Engineering building. And another one generated with a simulated environment with various weather conditions.

In Chapter 4, a practical method for incorporating existing policies with RL. For many control and automation systems, there are existing policies whether they are model-based, rule-based, or any other optimization methods. Most of them are more stable than deep-RL methods. We take advantage of both the existing policy for its robustness and stability and deep-RL for its scalability and learnability.

Chapter 5 transitions our focus from an offline to an online approach, aiming at the continuous improvement of the agent. This is achieved through the automatic fine-tuning of policy regularization, demonstrating the potential for further enhancements.

In Chapter 6, the dissertation concludes with a comprehensive discussion of the outcomes of our research, reflections on the broader implications, and suggestions for future directions in this field.

# Chapter 2

# Safe HVAC Control via Batch Reinforcement Learning

## 2.1 Introduction

Buildings account for 28% of the global carbon emissions [115], and HVAC (heating, ventilation, and air conditioning) systems account for the majority of building energy consumption[1]. Modern data-driven algorithms have the potential to improve the energy efficiency of traditional HVAC control algorithms. Here we focus on HVAC control in office buildings.

An office building is typically divided into multiple thermal zones, each of which can be controlled locally with a variable air volume unit. The control policy is based on sensor measurements (temperature, humidity, $CO_2$, airflow, etc) in the thermal zone. Rule-based

[1] https://www.eia.gov/energyexplained/use-of-energy/commercial-buildings.php

control (RBC) method is widely used to control the actuators [94], typically in conjunction with proportional-integral-derivative (PID) controllers [63, 25]. Such controls are interpretable but rely on experience and rules of thumb. It becomes challenging to develop and maintain a fine-grained RBC policy for dynamic environments. RBC is also a reactive algorithm as it changes the control settings based on the current measurements. We can improve the control performance if we can forecast the thermal environment characteristics.

We can predict thermal characteristics based on weather conditions, expected usage, and thermal insulation properties. In model-based approaches, thermal states of the building are simulated with simplified linear models for quick updates, and methods such as model predictive control (MPC) [10, 132, 5, 74, 85, 73] and fuzzy control [102, 15] are used to improve upon RBC policies. However, based on heating/cooling physics, we know that thermal evolution is non-linear with respect to indoor/outdoor conditions and depends on conditions such as orientation with respect to the sun, use of blinds, and wall insulation properties. Therefore, we can devise more accurate models to improve control performance further. Simulators such as EnergyPlus [18] and TRNSYS [53] have been designed to capture the thermal properties of a building. However, designing and calibrating such models for a large building requires significant time and expertise. With advances in sensing technologies and machine learning, data-driven models have become popular in recent research.

Reinforcement learning (RL) methods learn via direct interaction with the environment and thus have been studied extensively [40, 126, 136]. They are categorized into model-based RL (MBRL) [23, 81] and model-free RL (MFRL) [146, 17, 38] algorithms. MBRL requires the use of a simulator such as EnergyPlus [18], TRNSYS [53]. Without the pre-training offline, their nature to take exploratory control actions can cause occupant discomfort. MBRL relies on a thermal model learned from historical data, converges faster than MFRL methods. However, even with MBRL, the initial policy is worse than the existing control policy, and it can take weeks/months to improve and converge [24]. By contrast, batch RL can learn directly from historical data. Previous studies have shown that BRL methods can improve on existing policies [30] by exploiting the behavioral policy to identify actions that maximize the reward over an episode with TD-error update (Q-learning) while ensuring that the chosen actions do not deviate too far from the existing policy so the value estimation is more accurate. Typically, there is a hyperparameter to decide the learning trade-off between Q-learning or behavior cloning. Therefore, batch RL is a more efficient method for deployment when historical data is available. We are the first to explore the BRL deployment on real building HVAC control.

We design a BRL-based solution that effectively learns from available historical data without requiring the use of a simulator or explicit modeling of the HVAC system. Our framework guarantees safe system operations by avoiding random setpoint exploration that could damage the equipment and/or make occupants uncomfortable.

Our main contributions are summarized as follows:

- We propose and develop our framework, a simulation-free control algorithm for energy reduction and thermal comfort co-optimization. Our framework learns from existing historical data only, without requiring a simulator or complex modeling of the space.

- Our method Batch Constrained Munchausen deep Q-learning outperforms state-of-the-art BRL methods by penalizing policies that deviate too far away from the previous policy. It outperforms existing controls from the first day of deployment.
- We compare our framework with several state-of-the-art BRL methods. Our framework reduces energy consumption by 16.7% compared to the default control, which is 7.2% improvement over the state-of-the-art, while keeping thermal comfort during the entire evaluation period.

## 2.2 Background and related work

Our work on simulation-free framework for real-life multi-zone, multi-floor buildings build up following previous works.

### 2.2.1 Model Predictive Control

MPC methods use a model to forecast the outdoor and indoor conditions and optimize for a sequence of control actions that maximizes the given objective. MPC has been studied by several prior works for HVAC control. Aswani et al. [5] use learning-based MPC to control the room temperature to optimize energy consumption. Beltran et al. [10] use occupancy prediction models derived from occupancy data traces and minimize energy consumption while staying within the comfort bounds of the occupants. Maasoumy et al. [74] propose a model-based hierarchical control strategy that balances comfort and energy consumption. They linearize their thermal dynamics model around its operating point and use an LQR supervisory controller that selects the optimal setpoints for the lower level PID controllers. Privara et al. [85] interconnect building simulation software and traditional identification methods to avoid the statistical problems with data gathered from the real building. Winkler et al. [132] develop a data-driven gray-box model whose parameters are learned from building operational data. Together with weather forecast information, this data is fed into the framework to minimize energy costs while satisfying user comfort constraints.

Overall, the known issues of MPC are that it requires an accurate dynamic model, makes convexity assumptions, and the computation cost of computing each control decision is high [9]. RL solutions have been shown to overcome these limitations and outperform MPC methods [77], and the computation cost of a control decision is low as it only requires a neural network inference.

### 2.2.2 Reinforcement Learning in Building Control

#### Online RL Methods

Zhang et al. [144] jointly optimize visual comfort, thermal comfort, and energy consumption by training for $\sim 12K$ days in a simulator. OCTOPUS [23] co-optimizes HVAC, lighting, blinds, and window systems and needs $\sim 3.5K$ days of training. Valladares et al. [116] co-optimize thermal comfort and indoor air quality requiring $\sim 3K$ days of training. Nagarathinam et al. [77] train a multi-agent policy by taking into account waterside chiller control, and reducing convergence time to 2 years ($\sim$700 days) using domain knowledge-based pruning. DeepComfort [38] uses DDPG [66] to co-optimize thermal comfort and energy consumption with $10^4$ hour (∼417 days) for training.

MBBC [24] compares MBRL and MFRL methods with multi-zone control and shows that at least $10^4$ time steps are needed to converge. Zhang et al. [139] train in an online fashion to control airflow and temperature. They also take ∼100 days to converge. All prior works need a simulator or a data-driven model to predict the thermal dynamics. Zhang et al. [146] use A3C [75] on real building deployment with model pre-trained on a simulator. HVACLearn [81] proposes an RL-based occupant-centric controller (OCC) for thermostats using tabular Q-Learning with EnergyPlus simulator. Raman et al. [87] implement Zap-Q [22] with $\epsilon$-greedy exploration and compare the model with MPC methods using EnergyPlus. Lu et al. [71] compare on-policy and off-policy RL models with simulated air-conditioned buildings with data-driven models.

Online RL methods, either model-free or model-based, rely on exploration of the state-action space to improve the control policy. Model-free approaches are particularly data inefficient (months to years of convergence time), and therefore, require the use of a simulation model to learn a policy that can be practically deployed. But deploying such policies to a real building requires careful calibration of the simulation model, which is prohibitively time-consuming and expensive. Model-based methods are comparably data-efficient and can use a thermal dynamics model trained with historical data. However, even these methods require weeks to months of real-world interaction for convergence. The initial control policy performance is considerably worse than the existing rule-based policy [24, 77], and becomes a large impediment to adoption. To setup an EnergyPlus model, we need building-specific information, such as materials used to construct the building, that require consulting blueprints. Even after modeling with such details, a separate calibration step is required to ensure the accuracy of the model. Whereas for our reward function model, we used standard heat transfer equations and already available sensor data from the building management system. The reward function can be reused in a new building, whereas EnergyPlus will require redoing the work again. Without a model to simulate airflow, we use the readings and set points from the building management system. These are standard data points available in modern buildings, and our method can be reused as is in other buildings.

### Offline RL Methods

Offline methods have not been explored much yet in the building controls domain. GNU-RL [17] implements behavior cloning for HVAC control. In contrast to behavior cloning [95], where the agent simply learns to copy the behavioral policy with an ML model, the BRL method is able to learn from the existing data with Q-update and compensate for the lack of diversity in the buffer by perturbing the selected action with a perturbation network. BRL maximizes the values returned by selecting policy that improves upon the existing policy, rather than imitating it.

Previously, Ruelens et al.'s works focus on electricity cost optimization [92], demand response [91], and energy efficiency of heat pump [93] using fitted Q-iteration (FQI [89]). Vazques et al. [118, 119] balance comfort and energy consumption of a heat pump using FQI. Yang et al. [135] implement Batch Q-Learning for low exergy buildings. Our approach is closer to Wei et al.'s work [128], using a modified Q-learning algorithm, where they clip and shrink the reward value to control airflow for offline training. Unlike our method, the experiments are done in simulators, do not control zone temperature setpoint, and only consider temperature as a proxy for thermal comfort.

Algorithms such as FQI, Batch Q-learning, and Wei et al.'s DQN heuristic are all off-policy algorithms. Fujimoto et al. [36] show that off-policy methods exacerbate the extrapolation error in a completely offline setting. The errors occur because the Q-network is trained on historical data but exploratory actions yield policies which are different from the behavioral ones. They propose Batch Constrained Q-learning (BCQ) [36] which restricts selected actions to be close to those in the historical data and outperforms prior approaches. BCQ uses a Variational AutoEncoder (VAE) [51] to reconstruct the predicted actions given current states according to existing data.

BCQ is designed for complete offline, off-policy learning to penalize policies that are far from the behavioral policies in the replay buffer. We build upon the BCQ algorithm to further constrain new policy to be close to the previous one. We enforce this constraint through Kullback-Leibler (KL) divergence between the learned policy and historical policy [122]. We show that our algorithm performance is more stable than BCQ in our real building evaluation.

We use the existing dataset as the prior experience, since the rules are made by domain experts, its behavioral policy is *safe* cf. random initialized online policy. In this Chapter, we focus on the performance of the algorithm in the initial days (one week) of deployment.

## 2.3 Design of our framework

### 2.3.1 BRL-based Control Framework Setup

As shown in Fig. 2.1, we first obtain historical data and process them into a replay buffer containing the transition tuples. At each time step, the BRL model will randomly sample a mini-batch from the replay buffer and train the target networks with the transitions sampled. Periodically (according to the *eval_freq* in Alg. 1), we evaluate the trained agent's policy (the *select_action* function in Alg. 2) on real building zones to observe the states from our system's readings and calculate the reward. The average rewards over time are shown in Fig. 2.5.

We use the episodic formulation as this is the standard procedure in BRL literature [36, 70, 58]. In our formulation, the episode ends if the predicted thermal comfort is out of the thermal comfort range, i.e. the absolute value larger than 0.5. Therefore, the agent is trained for an arbitrarily long episode length as long as it does not impact comfort. If we use a fixed episode length such as 24 hours, the agent will optimize for that period. We use a time step of 9 minutes because that is the data-writing period for our building management system. We choose the minimum possible time step to minimize system response time and reduce any discomfort to occupants.

We represent the agent and its environment as a Markov Decision Process (MDP) defined by a tuple, $M_{\mathcal{B}} = (\mathcal{S}, \mathcal{S}', \mathcal{A}, P, R, \gamma)$, where $\mathcal{A}$ is the action space in the batch $\mathcal{B}$, $\mathcal{S}$ is the state space, $\mathcal{S}^{'}$ is the arriving state space where $\forall s' \in \mathcal{S}'$ corresponds to $s \in \mathcal{S}$ at a certain time step $t$ such that $s = s_t$, $s' = s_{t+1}$. $P(s'|s, a)$ is the transition distribution,

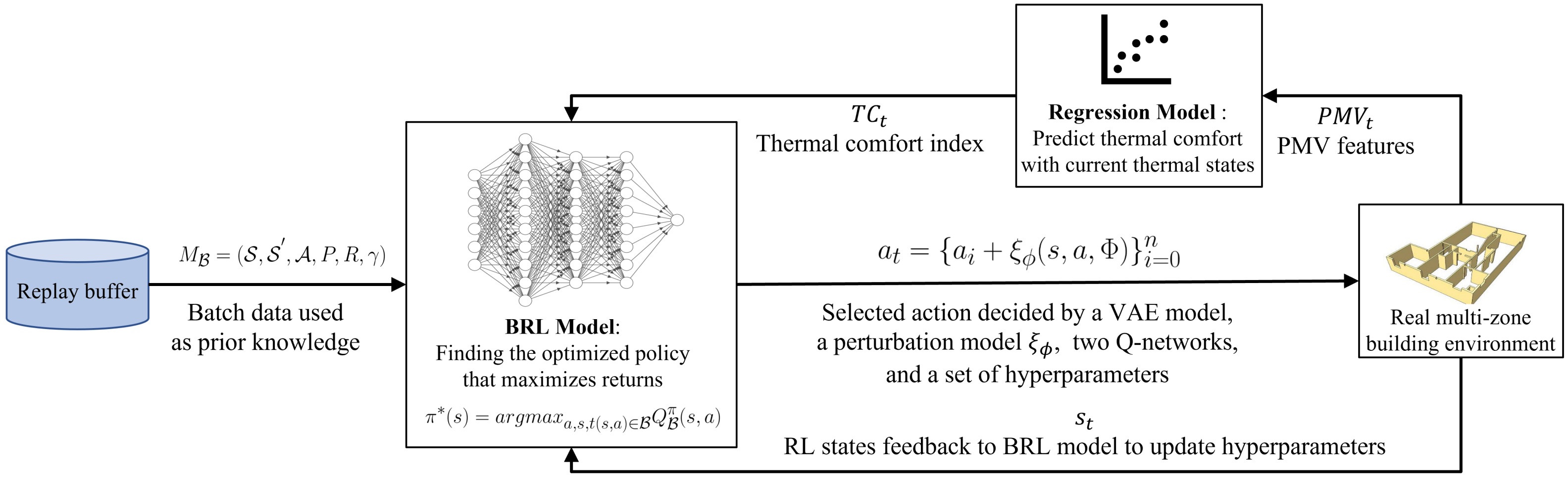

Replay buffer
$M_{\mathcal{B}} = (\mathcal{S}, \mathcal{S}^{'}, \mathcal{A}, P, R, \gamma)$
Batch data used
as prior knowledge
BRL Model:
Finding the optimized policy
that maximizes returns
$\pi^*(s) = argmax_{a,s,t(s,a)\in\mathcal{B}} Q^{\pi}_{\mathcal{B}}(s,a)$
$TC_t$
Thermal comfort index
Regression Model :
Predict thermal comfort
with current thermal states
$PMV_t$
PMV features
$a_t = \{a_i + \xi_\phi(s, a, \Phi)\}_{i=0}^{n}$
Selected action decided by a VAE model,
a perturbation model $\xi_\phi$, two Q-networks,
and a set of hyperparameters
$s_t$
RL states feedback to BRL model to update hyperparameters
Real multi-zone
building environment

**Figure 2.1**: Overview of our batch reinforcement learning model selects actions that co-optimize thermal comfort for occupants and energy consumption of HVAC system.

$R(s, a)$ is the reward function, and $\gamma \in [0, 1)$ is the discount factor. The goal of our BRL model is to find an optimal policy $\pi^*(s) = \text{argmax}_{a \ s.t.(s,a) \in \mathcal{B}} Q^{\pi}_{\mathcal{B}}(s, a)$, which maximizes the expected accumulative discounted rewards.

More specifically, we model the following:

- *State*: We use the following attributes for the RL process to evaluate the policy: indoor air temperature, actual supply airflow, outside air temperature, and humidity. These states include the features needed for thermal comfort estimation $s_t^{TC}$ and those that represent the responses of actions as RL states $s_t^{RL}$.
- *Action*: We control two important parameters, namely, zone air temperature setpoint ($a_t^{ZNT}$) and actual supply airflow setpoint ($a_t^{Sup}$). Both are in continuous space and the action spaces are normalized in the range of $[-1, 1]$.
- *Environment*: Real building HVAC zones across three different floors. Every room is a single HVAC zone, and all these rooms are used as lab space and work office.
- *Reward*: We monitor the thermal states of the space as well as the thermal comfort index predicted by a regression model, and then make control decisions with the actions selected by the BRL model. Our reward function penalizes high HVAC energy use and discourages a large absolute value of the thermal comfort index, which indicates discomfort to occupants. Our reward function at time step $t$ is:

$$R_t = -\alpha |TC_t| - \beta P_t, \tag{2.1}$$

where $\alpha$, $\beta$ are the weights balancing between different objectives and could be tuned to meet specific goals, $TC_t$ is the thermal comfort index at time $t$, $P_t$ is the HVAC power consumption at time $t$. We compute $P_t$ attributed to a thermal zone using heat transfer equations [6]. The DRL agent co-optimizes HVAC energy reduction and occupants' thermal comfort.

### 2.3.2 Thermal Comfort Prediction

To calculate the thermal comfort level for our reward function, we adopt the widely used predicted mean vote (PMV) [27] measure as our thermal comfort index. In this metric, there are degrees of satisfaction, ranging from $-3$ (cold) to 3 (hot), where PMV within the range from $-0.5$ to 0.5 is considered thermal-comfortable.

We adopt the ASHRAE RP-884 thermal comfort data set [21] and train a simple gradient boosting decision tree model [48] to predict the thermal comfort by taking the current thermal states given by our building system in real-time. We show the evaluation of the effectiveness in Fig. 2.2 with such a simple GBDT-based thermal comfort index.

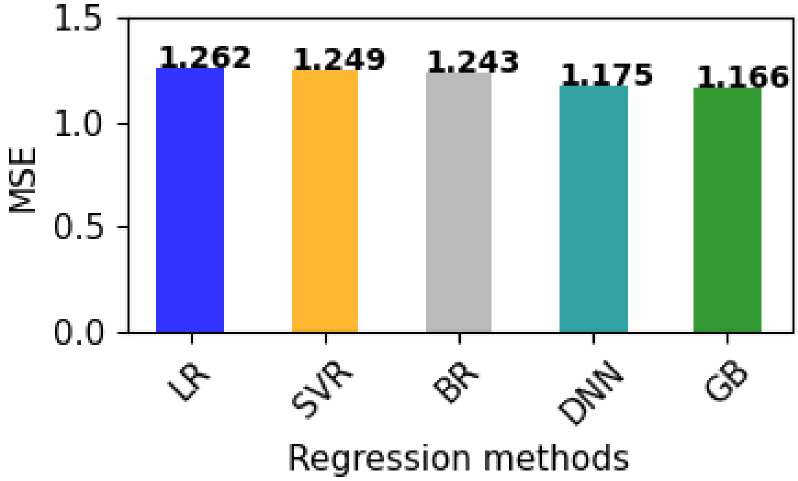


**Figure 2.2**: Performance comparison of regression models for predicting thermal comfort based on PMV

### 2.3.3 Batch Reinforcement Learning for Control

We take a BRL-based method, namely, batch-constrained deep Q-learning (BCQ) [36] as our foundation and make improvements on it. BCQ is a pure offline, off-policy RL method that avoids the extrapolation errors induced by the incorrect value estimation of out-of-distribution actions selected out of the existing dataset.

As illustrated in Fig. 2.1, for each time step $t$, we obtain state information from the sensors in the building. To only calculate the reward but not update the models. BCQ first samples a mini-batch of data (the size of the mini-batch is set as a hyperparameter) from the entire set of historical data. Then, it trains a parametric generative model $G_\omega$, a conditional VAE on the batch to model the distribution by transforming an underlying latent space. The encoder $E_{\omega_1}(s, a)$ takes a distribution of state-action pairs and outputs the mean $\mu$ and standard deviation $\sigma$ of a Gaussian distribution $\mathcal{N}(\mu, \sigma)$. A latent vector $z$ sampled from the Gaussian is passed to the decoder $D_{\omega_2}(s, z)$ which outputs an action. The loss function of VAE consists of two parts: reconstruction loss and the KL regularization term $\lambda \mathcal{L}_{KL}$.

$$\mathcal{L}_{recon} = \sum_{(s,a)\in\mathcal{B}} (D_{\omega_2}(s, z) - a)^2, z = \mu + \sigma \cdot \epsilon, \epsilon \sim \mathcal{N}(0, 1) \ \mathcal{L}_{_{KL}} = D_{_{KL}}(\mathcal{N}(\mu, \sigma)||\mathcal{N}(0, 1)),$$

$$\mathcal{L}_{_{VAE}} = \mathcal{L}_{recon} + \lambda \mathcal{L}_{_{KL}}.$$

VAE here aims to produce only actions that are similar to existing actions in the batch given the current state. The purpose of the perturbation model $\xi_\phi(s, a, \Phi)$ is to increase the diversity of seen actions, it adjusts the value of the selected action $a$ in the range of

$[-\Phi, \Phi]$, (where $\Phi$ is the max perturbation). It could compensate for the lack of diversity in the batch data, as a trade-off of inaccurate value estimation. By adjusting the hyper-parameters $n$ and $\Phi$, it could behave similarly to behavior cloning with $n = 1$ and $\Phi = 0$, or similarly to traditional Q-learning when $n \to \infty$ and $\Phi \to a_{max} - a_{min}$.

$$\phi \leftarrow \underset{\phi}{\operatorname{argmax}} \sum_{(s,a)\in\mathcal{B}} Q_{\theta_1}(s, a + \xi_\phi(s, a, \Phi)), a \sim G_\omega(s).$$

At the core of BCQ is the value estimation networks, a pair of Q-networks $Q_{\theta_1}(s, a)$ and $Q_{\theta_2}(s, a)$. By taking a weighted minimum between the two values as a learning target $y$ for both networks. On the other hand, for the actor-network, at first, $n$ actions are sampled with respect to the generative model, and then adjusted by the target perturbation model, before being passed to each target Q-networks for updates:

$$y = r + \gamma \max_{a_i} \left[ \lambda \min_{j=1,2} Q_{\theta'_j}(s', \tilde{a}_i) + (1 - \lambda) \max_{j=1,2} Q_{\theta'_j}(s', \tilde{a}_i) \right], \tag{2.2}$$

where $r$ is the reward, $\gamma$ is the discount factor, $\lambda$ is the minimum weighting in double-Q learning, $\theta_{j=1,2}$ are weights of the two critic Q-networks.

We propose an improvement on the BCQ algorithm, called Batch Constrained Munchausen RL (BCM), that encourages the agent to update the policy close to the previous one using a regularization term in the Q-update. With respect to other aspects, BCM inherits BCQ's characteristics and acts as an intermediate state of behavior cloning and Q-learning.

The idea of the BCM algorithm is the following: we adopt the regularization term in Munchausen RL (M-RL) [122] which penalizes the policies which deviate far from the previous policy with Kullback-Leibler (KL) divergence [121, 56]. M-RL utilizes the current policy as one of Q-update's learning signals. $KL(\pi_1||\pi_2) = \langle \pi_1, \ln \pi_1 - \ln \pi_2 \rangle$. The other term added in M-RL is the entropy term which penalizes the policies that are too far away from the uniform distribution, where $\mathcal{H}(\pi) = -\langle \pi, \ln \pi \rangle$. In offline settings, this term does not help improve the Q-update since we cannot accurately estimate uniform policy if we have only static data. We do not encourage exploration as the online mode in the original M-RL settings by adding the entropy term. Our problem is focused on conservative and safe policies exclusively selected from the batch with a small amount of perturbation. It helps to avoid the lack of diversity within state-action visitation in the batch distribution.

$$y = r + \alpha_m \left[\tau_m \ln \pi_{\hat{\theta}}(a_t|s_t)\right]_{l_0}^{0} + \gamma \max_{a_i} \left[\lambda \min_{j=1,2} Q_{\theta'_j}(s', \tilde{a}_i) + (1-\lambda) \max_{j=1,2} Q_{\theta'_j}(s', \tilde{a}_i)\right], \tag{2.3}$$

where $\pi_{\hat{\theta}} = softmax(\frac{Q_{\hat{\theta}}}{\tau})$, the target Q after soft clipping in double Q-learning, $\alpha_m$ is the M-RL scaling parameter, $\tau_m$ is the entropy temperature parameter, and $l_0$ is the clipping value minimum since the log-policy term is not bounded and can cause numerical issues if the policy becomes too close to deterministic. We replace $\tau \ ln \ \pi(a|s)$ by $[\tau \ ln \ \pi(a|s)]_{l_0}^{0}$, where $[\cdot]_x^y$ is the clipping function. The other added term in the original M-RL algorithm is the entropy term which encourages policies to be close to uniform distribution. We do not use it as it is not applicable for offline settings [122]. Once we choose the action using BCM, we adjust the corresponding setpoints through a building operating system (BOS) [130, 55]. The environment reflects the real response of action applied with a time delay $d$, so our framework waits for $d$ to get data $s_t$ from the sensors. Also, a PMV feature vector $PMV_t$ is fed into the regression model for thermal comfort prediction. According to the prediction of regression model $TC_t$ and the RL states $s_t$, we calculate the reward using Eq.(3.1). We repeat this process until reaching the maximum number of time steps $T$. Details of the HVAC control via BCM algorithm are described in Algorithm 1.

## 2.4 Evaluation

### 2.4.1 Data Collection and Pre-processing

The data we use from all the sensors and control points are recorded every 9 minutes via a BOS (Building Operation System). We obtain data for an entire year, from the beginning of July 2017 to the end of June 2018 of fifteen rooms across three different floors in a building, the details of which are in the appendix. The batch for each floor, or the *buffer*, contains around $200K$ transitions (2F:$\sim 260K$, 3F:$\sim 193K$, 4F:$\sim 172K$), and it might differ from one to another due to varied system maintenance duration throughout the year. Since the rooms on the same side of a floor often share similar thermal dynamics, we thus create batch data for each floor to ensure that the replay buffer reflects each variable air volume (VAV)'s thermal dynamics precisely. We set each room to its maximum occupancy, which is obtained from our campus facility information management system, and in evaluation, we assume full occupancy the entire time for the most strict condition. We could easily modify the problem formulation by taking occupancy into account in both our policy and reward function. The airflow CFM (cubic feet per minute) needed is just multiplied by the number of people in the room. However, at this moment we have no occupancy sensor data, so we assume the most strict condition of full capacity. We standardize our actions

**Algorithm 1:** HVAC control via our framework

```
Input  : Batch data $\mathcal{B}_f$ for a certain floor $f$, time horizon $T$, floor set $\mathcal{F}$,
         zone/room set $\mathcal{Z}$, and delayed response time $d$, target network update
         rate $\tau$, mini-batch size $b$, max perturbation to selected actions $\Phi$,
         number of sampled actions $n$, minimum weighting $\lambda$, evaluation
         frequency $eval\_freq$, M-RL scaling factor $\alpha_m \in [0, 1]$, and entropy
         temperature parameter $\tau_m$
Output: Reward, next state, and action selected by BCM
Initialize: HVAC Environment $Env$, RL agent $BCM$
$d_a = dim(a), d_s = dim(s)$;
for $f \in \mathcal{F}$ do
    $BCM_f = BCM(d_s, d_a, \gamma, \tau, \lambda, \phi, \alpha_m, \tau_m)$;
    for $z \in \mathcal{Z}$ do
        0 ← t;
        while $train\_iteration < T$ do
            $BCM_f.train(\mathcal{B}_f, b, n)$;
            if $t\%eval\_freq == 0$ then
                $s_t^z = Env^z.getThermalState(t)$;
                $TC_t^z = Env^z.getPredictedTC(s_t^z)$;
                $a_t^z = BCM_f.select\_action(s_t^z)$;
                $s_{t+1}^z, r_t^z = Env^z.step(a_t^z, s_t^z, d)$;
                $t += 1$ ;
```

in a batch to the range of $[-1, 1]$ as a standard procedure in the RL setup. For each action sample $a_i$, it is converted to $z_i$ such that $z_i = (a_i - \mu)/s$, where $\mu$ is mean, $s$ is the standard deviation of the batch. In the replay buffer, there are several main matrices required: action $\mathcal{A}$, state $\mathcal{S}$, next state $\mathcal{S}'$, reward $\mathcal{R}$ (calculated with our thermal comfort prediction model, power consumption, and RL states), index $\mathcal{I}$ (which records the indices as time stamps), and episode terminal status $\mathcal{N}$ (it labels if an episode is terminated or not—in our setting when the predicted thermal comfort metric does not satisfy the criteria, i.e. $|PMV| > 0.5$, the episode is considered as terminated). To summarize, the batch data is a set consisting of the above-mentioned matrices, i.e. $\mathcal{B} = \{\mathcal{A}, \mathcal{S}, \mathcal{S}', \mathcal{R}, \mathcal{I}, \mathcal{N}\}$.

### 2.4.2 Thermal Comfort Prediction

We compare five different regression models for predicting thermal comfort, namely Linear Regression (LR), Support Vector Regression (SVR), Bayesian Regression (BR), Deep Neural Network (DNN), and Gradient Boosting (GB) (Fig. 2.2). The input features of the models are zone air temperature, humidity, mean radiant temperature (MRT), air velocity, metabolic rate (Met), and clothing insulation (Clo). We set the clothing level as ”typical summer indoor clothing”. Metabolic rate is set as ”typing” reflecting the most common activities where the zones evaluated are all student lab and office spaces. There are in total 30650 data points with complete feature information in the ASHRAE RP-884 thermal comfort data set [21] we adopted for evaluation. All models are trained and tested with 10-fold cross-validation. Hyperparameter optimization is conducted via either grid search or Bayesian optimization. According to Fig. 2.2, the best model is the gradient boosting tree [48] with an MSE of 1.147, which supports our choice of GB-based model to predict thermal comfort index for the RL reward function. It is reasonable that the gradient boosting method outperforms the deep learning counterpart on tabular data because of selection bias and hyperparameter optimization [103] The MSE metrics reported are averaged with 3 runs for each model.

### 2.4.3 Importance of Airflow Control

Few prior works quantitatively study the importance of airflow control in maintaining occupants' thermal comfort. Almost all research focuses on temperature and humidity control for occupants' thermal comfort. Here, we empirically analyze how airflow impacts thermal comfort based on the PMV (Predicted Mean Vote) features.

We conduct analysis via mutual information-based regression. Between two random variables $(X, Y)$, the dependency of these two variables, which is a non-negative value, is calculated as:

$$I(X;Y) = \int_y \int_x p_{(X,Y)}(x,y) log \left( \frac{p_{(X,Y)}(x,y)}{p_X(x)p_Y(y)} \right) dx\, dy,$$

where $p(X, Y)$ is the joint probability density function of $X$ and $Y$, and $p_X, p_Y$ are the corresponding marginal density functions. It is equal to zero if and only if two random variables are independent, and higher values mean higher dependency [90]. Fig. 2.3 indicates that air velocity is the second most important factor after air temperature and mean radiant temperature (MRT) (here we approximate MRT with air temperature [20]). Thus, by controlling zone air temperature and airflow (air velocity can be converted to airflow rate with room area), we control the two most important features affecting occupants'

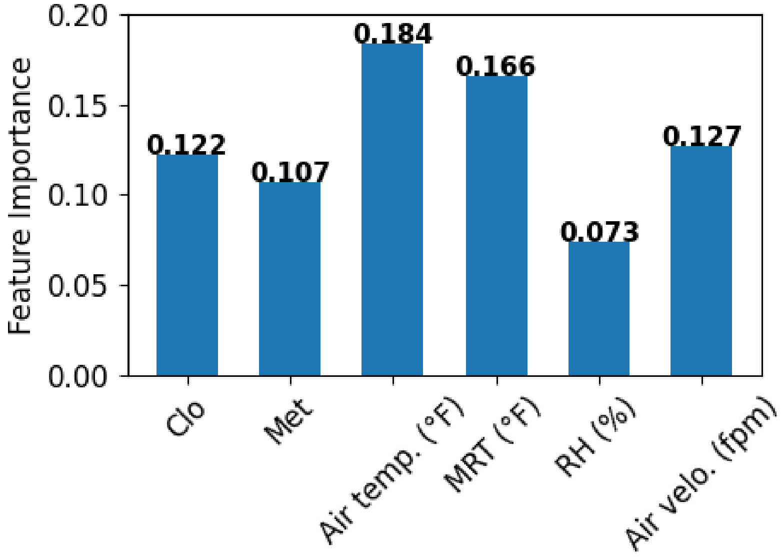


**Figure 2.3**: Importance of feature to thermal comfort via mutual information regression analysis. The features are clothing level (Clo), metabolic rate (Met) indoor air temperature (Air temp.), mean radiant temperature (MRT), relative humidity (RH), and air velocity (Air velo.).

thermal comfort.

### 2.4.4 Preliminary Experiments

We first investigate how BRL methods are compared with online RL methods, we compare these BRL methods with the state-of-the-art online RL methods: TD3 [34] and DDPG [66]. Our approach is to build a data-driven simulator environment with two VAEs (Fig. 2.4). The first one is for predicting the RL and thermal comfort states. The second one is to predict the power/energy consumption. These two VAEs function as the thermal states simulator.

We evaluate with 200 episodes and the evaluation frequency is every five time steps. We run each algorithm with three randomly initialized initial conditions in the range of our dataset. As we see in Fig. 2.4, the solid line is the average of these three runs, and the half-transparent regions indicate the range of these runs. The results show that the performance ranking among these BRL methods: PQL>BEAR>BCQ>BCM>DDPG>TD3.

While BRL methods reach a stable state, online RL methods TD3 and DDPG are still exploring new policies, and thus yield a continuously declining performance in a short period of time. These BRL methods (details of PQL and BEAR are elaborated in 2.4.5) learn exclusively from the batch provide stable, and safe policies. The reason why performance is constant is that in the simulation environment the responses of the system are deterministic which is different from the real building environments. (Fig. 2.5) In real building systems, the responses are stochastic.

### 2.4.5 Baseline Methods

#### State-of-the-art BRL Methods

After BCQ was proposed, there are several studies outperforming it in the OpenAI Gym [13] simulation environments. We implement these methods as baselines to be compared with BCM.

- Batch Constrained Deep Q-Learning (BCQ) [37]: BCQ is one of the pioneering works of offline RL, it is an actor-critic method that incorporates a Variational AutoEncoder (VAE) to reconstruct the selected action based on the current state.

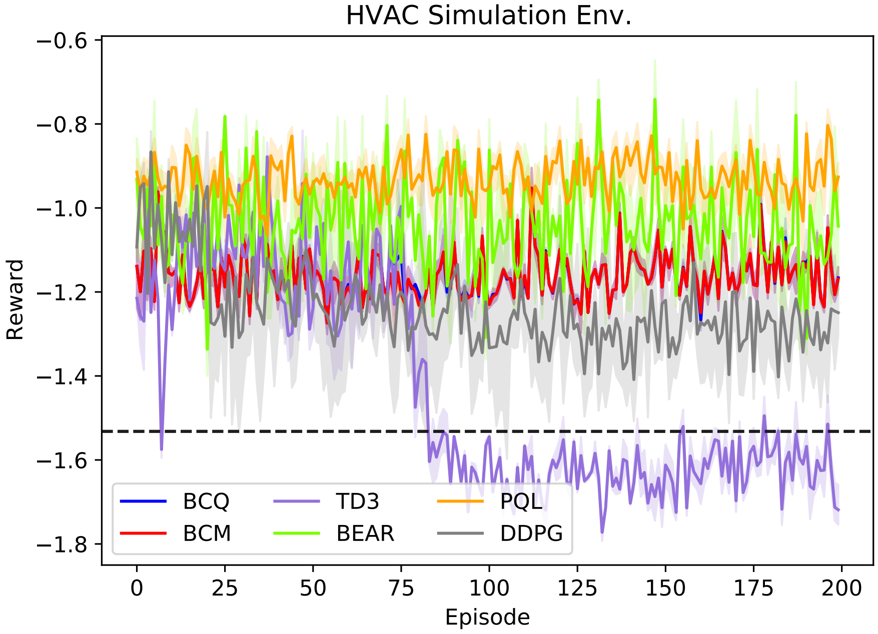


**Figure 2.4**: Performance comparison with VAE simulators

The key components of BCQ include:

- Perturbed Model: It uses a perturbed version of the policy to generate diverse and realistic action samples.
- Action Selection: Actions are selected based on a combination of value estimates and their likelihood under the behavior policy.
- Action Conditioning: The policy is trained to output actions conditioned on the state and a latent variable, encouraging exploration within the batch distribution.

- TD3 (Twin Delayed Deep Deterministic Policy Gradient) The TD3 (Twin Delayed Deep Deterministic Policy Gradient) algorithm enhances the DDPG (Deep Deterministic Policy Gradient) method to address its inherent challenges such as overestimation bias and training instability. TD3 introduces three main improvements to stabilize and improve performance. First, it uses a pair of critics (twin critics) to provide more accurate value estimates by taking the minimum value between the two, which reduces overestimation bias. Second, it delays the policy updates in comparison to the critic updates, ensuring that the critics are better trained before the policy is updated, which helps in stabilizing the training process. Third, it adds noise to the target action during value estimation to smooth the value function and prevent the policy from overfitting to specific actions. These enhancements make TD3 more robust and efficient, resulting in superior performance compared to DDPG in various continuous action space environments.

- DDPG (Deep Deterministic Policy Gradient) The DDPG (Deep Deterministic Policy Gradient) algorithm is an actor-critic method specifically designed for environments with continuous action spaces. In DDPG, the actor network outputs deterministic actions directly, while the critic network evaluates these actions. The actor is trained using the deterministic policy gradient, which is derived from the critic's Q-values. DDPG uses experience replay to break the correlation between consecutive experiences and stabilize training, as well as a target network to reduce the risk of divergence by providing a stable target for the critic network. While DDPG has shown strong performance in various tasks, it suffers from overestimation bias and instability during training, which led to the development of more advanced methods like TD3 to address these shortcomings.

- Bootstrapping Error Accumulation Reduction (BEAR) [58]: BEAR identifies bootstrapping error as a key source of BRL instability. It is due to the bootstrapping of actions that lie outside of the training data distribution. The algorithm mitigates the out-of-distribution action selection by searching over the set of policies that is akin to the behavior policy. BEAR's ultimate goal is to search over the set of policies $\Pi$, which shares the same set of values that the random variable can take on as the behavior policy. Its performance is outstanding with the medium-quality static dataset (medium-quality means by training an agent with half the amount of time steps cf. expert RL agent/human expert or when the agent is trained to yield half the average return cf. the expert agent).

- Pessimistic Q-Learning (PQL) [70]: While BRL yields a new policy other than those in the batch, it might visit states and actions that are outside the distribution of the batch data. In addition, function approximation with a limited number of samples leads to overly optimistic estimates. PQL thus uses pessimistic value estimates in the low-data regions in the Bellman optimality equation as well as the evaluation backup. It can yield more adaptive and stronger guarantees when the concentrability assumption does not hold. PQL learns from policies that satisfy a bounded density ratio assumption akin to the on-policy policy gradient methods. The approach of PQL to improve from BCQ's architecture is that they add a state-VAE to predict the arriving state given the current state-action pair, filtering state-action distribution

$\tilde{\mu}(s, a)$ instead of $\tilde{\mu}(s|a)$. The filtration is implemented by setting a hyperparameter $b$ as the $2^{nd}$ percentile of the state-VAE Evidence Lower Bound (ELBO). If the ELBO is larger than $b$ then Q-update is executed, otherwise, it is not executed.

#### Comparison Methodology

We run each algorithm in a single room on each floor in the same week so that outside air temperature (OAT) is the same. For instance, in one week we run our BCM in rooms in the same *stack* on different floors, e.g. 2144, 3144, and 4144, and at the same time a different BRL algorithm, e.g. BEAR, is running in rooms in a different adjacent stack, say, 2146, 3146, and 4146. In each room, we run the algorithm for $1,000$ time steps, which is about one week. To reduce performance variations, we evaluate each algorithm in three different rooms (one room from each floor: 2F, 3F, and 4F). These rooms have the same functionality (lab or office spaces) and are of roughly the same size and occupancy capacity. The entire evaluation time of all the experiments is from September 28th to October 19th, 2021.

Appendix A.1.5 lists the hyperparameters for each method.

### 2.4.6 Results and Analysis

#### Reward Comparison

Fig. 2.5 shows the evaluation results of each algorithm, where each solid line is the average reward of all runs for the same method; semi-transparent bands represent the range

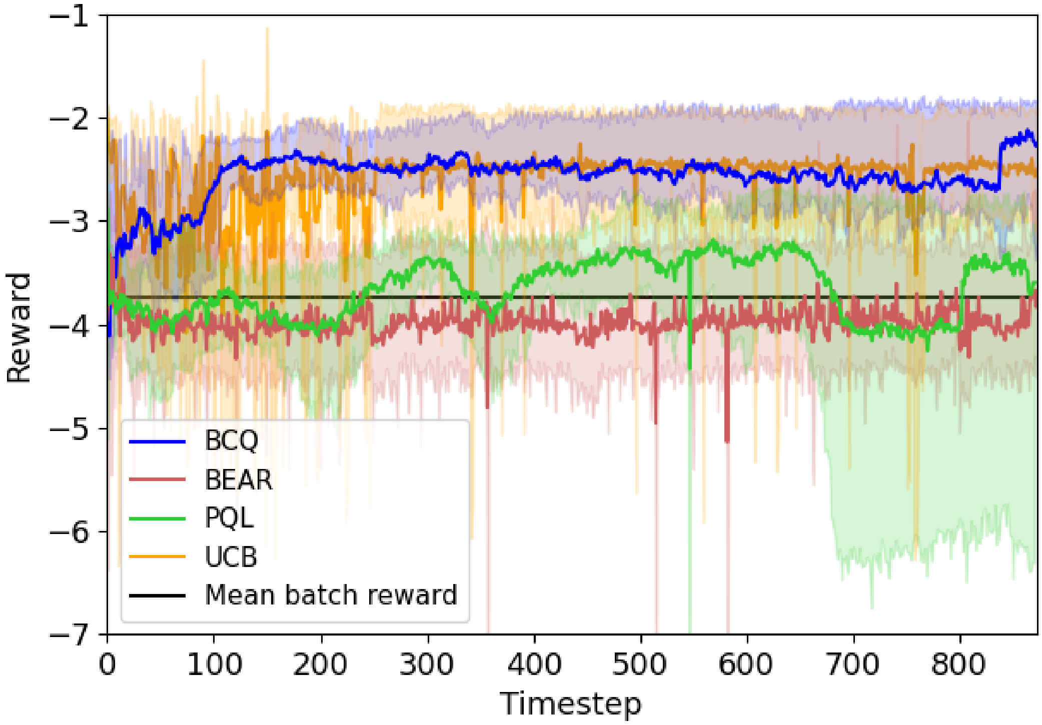


**Figure 2.5**: Reward comparison of various algorithms

of all runs for a particular algorithm. And gray dotted vertical lines indicate 00:00AM of each day. The horizontal black dotted line is the average reward in the buffer. It shows that BCM outperforms other methods by providing a relatively stable learning curve.

PQL constrains the Bellman update over state-action pairs that are sufficiently covered by the conditional probability of action given state when generating the data. It adds a state-VAE and a statistical filtration over BCQ's architecture with pessimistic value approximation, which might overkill near-optimal policy that is without enough visitation, however, as time evolves, PQL gradually learns better. BEAR is only guaranteed to outperform BCQ on medium-quality data sets collected from a partially trained policy – a middle ground between optimal policy and random policy. However, in our case, the replay buffers are closer to the data generated with expert policy. This explains the outcomes in a reasonable way. BCQ, as an ablation version of our BCM algorithm, yields a comparable performance as BCM but fails to keep a stable outcome due to the lack of a strong learning signal.

The comparison between algorithms in our experiments is distinct from the results shown in the original papers, where PQL outperforms BCQ and BEAR in two out of the three simulated environments. By contrast, on our real building HVAC system, BCQ provides a more stable and continuously improving performance than the other two BRL methods. This is because all those experiments were conducted in simulation environments where data are effectively unlimited, consequences for poor actions are non-existent, and system dynamics are clean and often deterministic [26]. However, in real-world problems, systems are stochastic and non-stationary. It is not guaranteed that these algorithms would behave the same or similar to simulated cases in these settings.

### Energy Consumption and Thermal Comfort Comparison

Outside Air Temperature (OAT) is a key factor affecting zone temperature; therefore, it affects both thermal comfort and energy usage of the HVAC system. It is thus reasonable to compare energy consumption during baseline time periods with the most similar OAT trend to the period during which these BRL methods are evaluated. To do so, we adopt Dynamic Time Warping (DTW) [12] to find historical weeks with similar OATs, as DTW is a widely used method to measure the similarity of time-series data of different lengths. In addition to considering the "shape" of historical OAT, we also consider the mean OAT

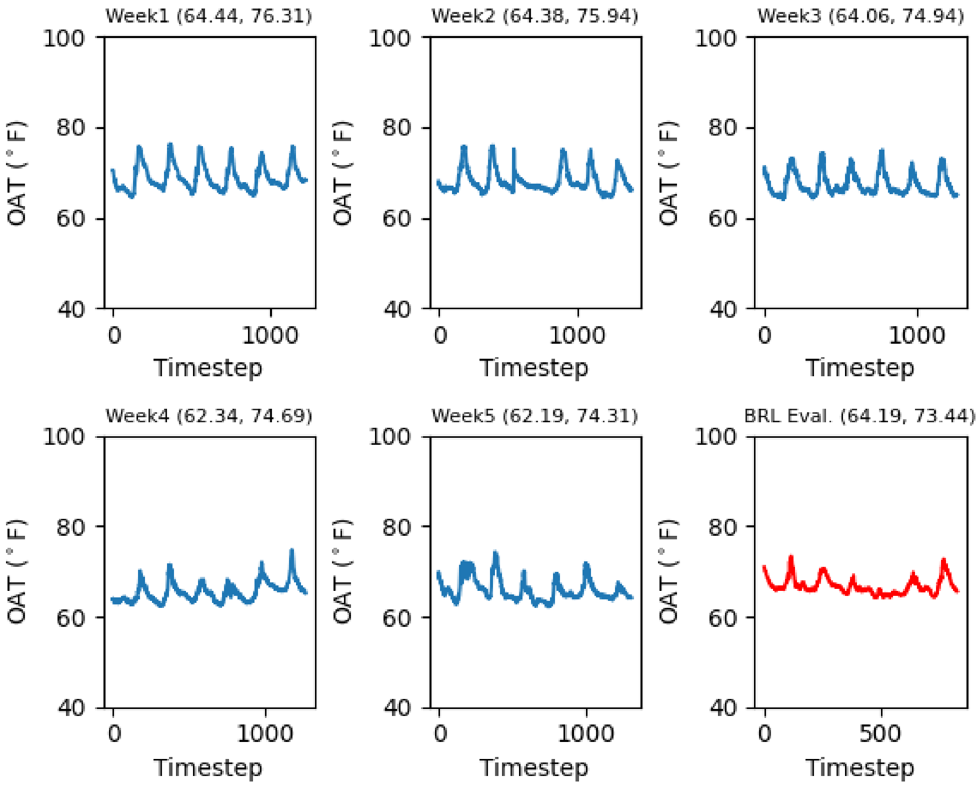


**Figure 2.6**: Outside Air Temperature (OAT) Comparison

difference between our experiment time period and historical weeks. In summary, we find historical time periods whose OAT trend is similar *and* with close average weekly OAT to our experiment week. Fig. 2.6[2] shows an example of historical weeks found using the above metrics. In this figure, a tuple of (*min, max*) OAT is labeled on top of each week's OAT data.

Once we have the top-5 weeks with the most similar OAT trend to our experiment period, we compare all methods and estimate energy consumption and thermal comfort.

In Fig. 2.7, we normalize the historical energy use to one as reference. BCM consumes

[2] We find the top-5 most similar weeks regarding OAT to our experiment week (last figure) for evaluating energy consumption and thermal comfort.

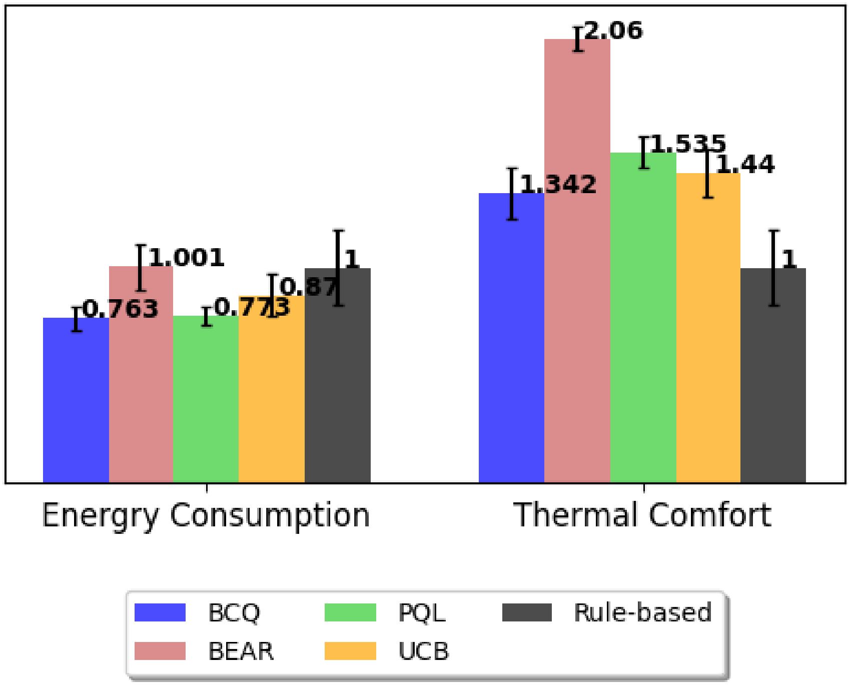


**Figure 2.7**: Energy consumption and thermal comfort comparisons among different control methods

the least energy compared with other methods. A 16.7% of energy consumption reduction is achieved, and BCQ also outperforms RBC by 9.5%. On the other hand, the occupants' thermal comforts are shown in real average absolute values. The standard deviations (marked as error bars) of all BRL methods are smaller than their historical counterparts.

We also examine the thermal comfort during the entire time period for every experiment and keep track of changes and violations as time evolves. Fig. 2.8 is an example showing that BCM maintains thermal comfort level persistently during the entire evaluation time period.

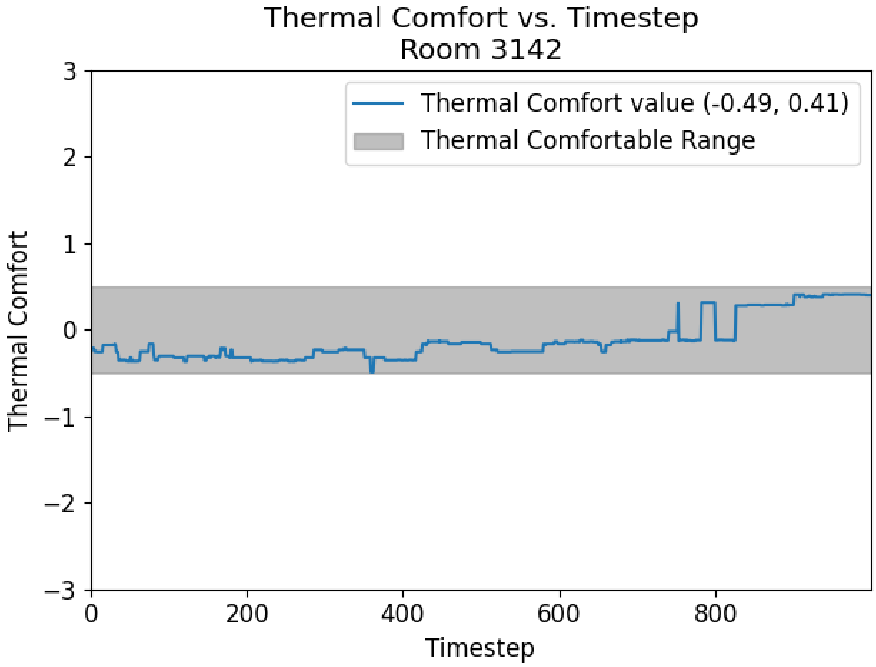


**Figure 2.8**: Thermal comfort achieved by our BCM model during evaluation

### 2.4.7 Sensitivity Analysis

#### Perturbation to Action

In our main evaluation, we used $\Phi = 0.05$, which is the parameter controlling the degree of perturbation applied to selected actions. To inspect how perturbation impacts the performance of BCM, we evaluate two different values of 0.1 and 0.2 for $\Phi$. The result in Fig. 2.9 indicates that for $\Phi = 0.1$, on average, does not yield a higher reward than $\Phi = 0.05$. For $\Phi = 0.2$, it cannot learn efficiently until around 700 time steps due to too large the range of action spaces to select from. In our buffer, there is enough diversity since it is extracted with an entire year of data. Thus, we choose $\Phi = 0.05$ in our main experiment.

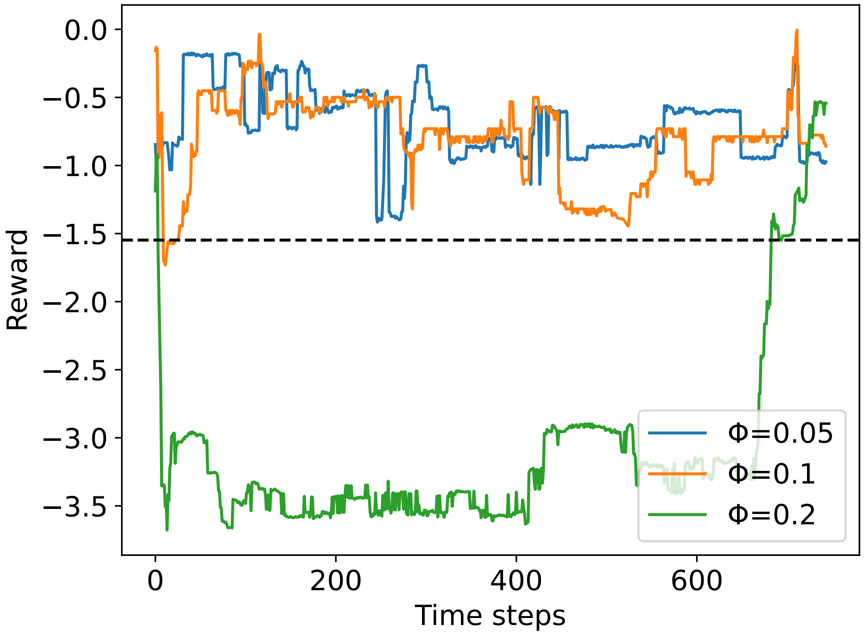


**Figure 2.9**: Effect of perturbation to selected actions

### Amount of Data

We randomly sample data points by a fraction of $\{\frac{1}{10}, \frac{1}{100}, \frac{1}{1000}\}$ and evaluate rooms on the same floor in the same week to observe the impact. Fig. 2.10 shows the information loss from smaller buffer data. E.g. for the $\frac{1}{1000}$ one, it hardly reaches the average of the original buffer. For the $\frac{1}{10}$ and $\frac{1}{100}$ cases, they show comparable performances but have difficulties being consistent.

### Diversity of Batch Data

Originally, we use the thermal states of a set of rooms/zones from an entire year as our batch data. Intuitively, a replay buffer containing data from the same season as our evaluation period might be more suitable because of the similar seasonal weather condition.

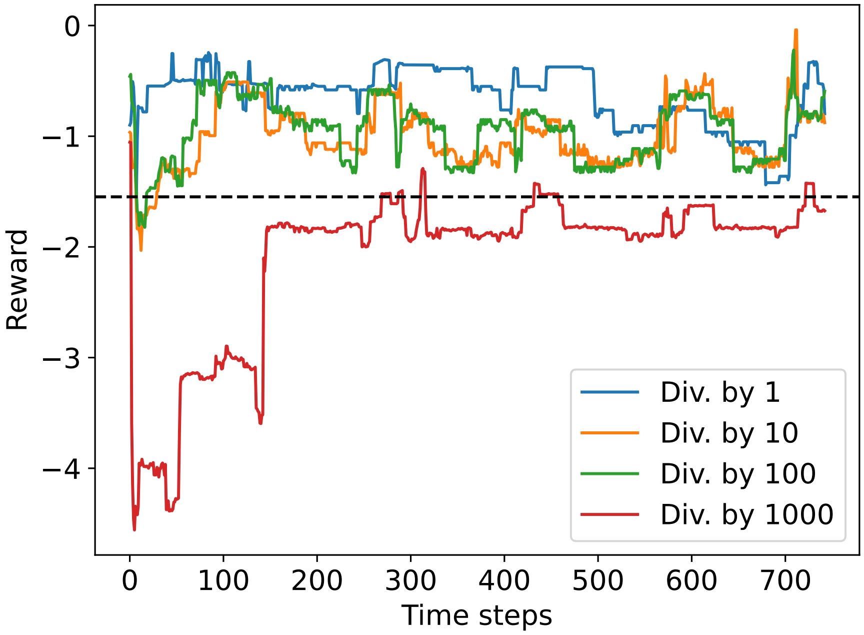


**Figure 2.10**: Effect of buffer data size

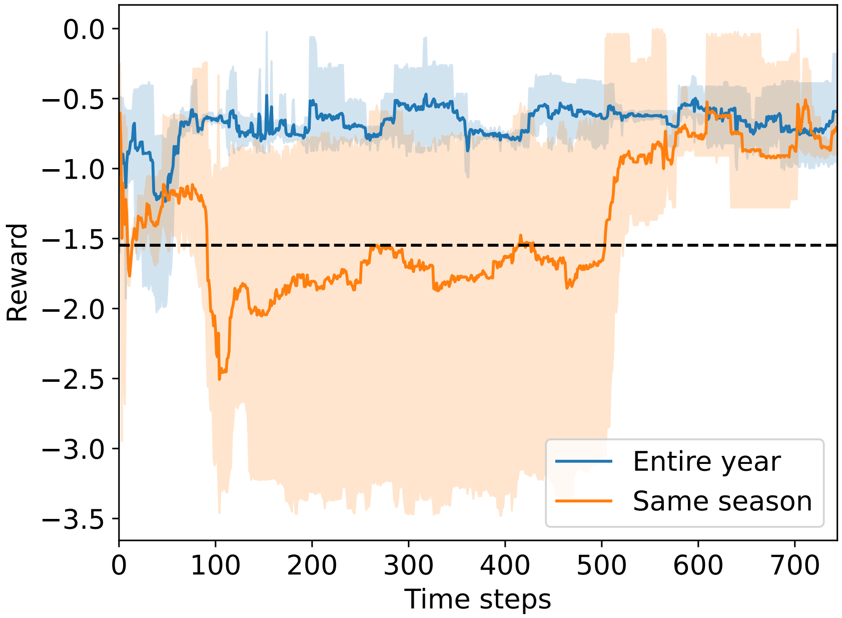


**Figure 2.11**: Same Season vs. Entire Year

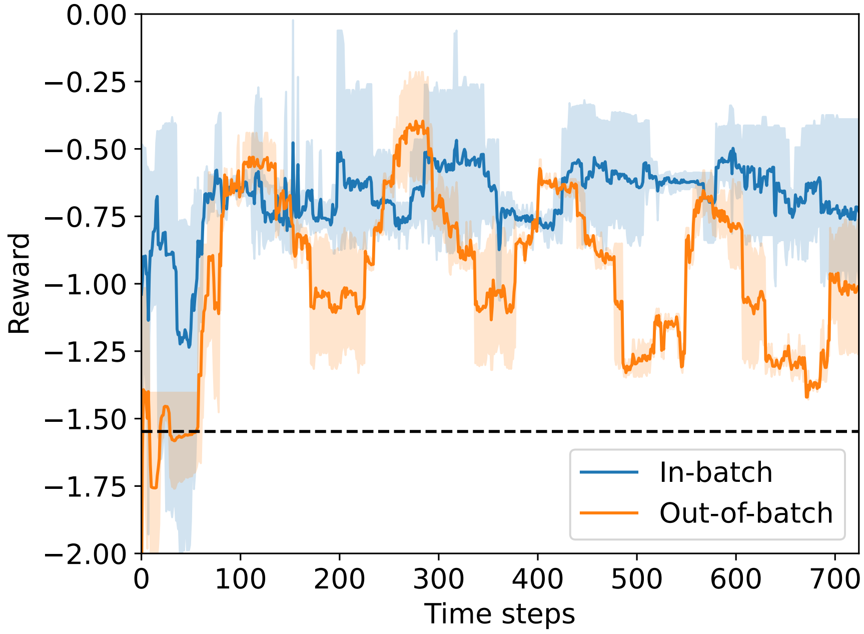


**Figure 2.12**: Out-of-batch (OOB) vs In-batch

Thus, we use only the data from the same season as an ablation.

Fig. 2.11 shows that a batch of the entire year's data produces better performance than only using the same season. This is becoming a narrower distribution of state-action visitation in a single season cannot update the Q-value as accurately as an entire year's data could. Incorrect Q-value estimation would lead to a lower return. In summary, it is essential to ensure enough state-action visitation diversity in the batch data, in order to estimate the value more accurately.

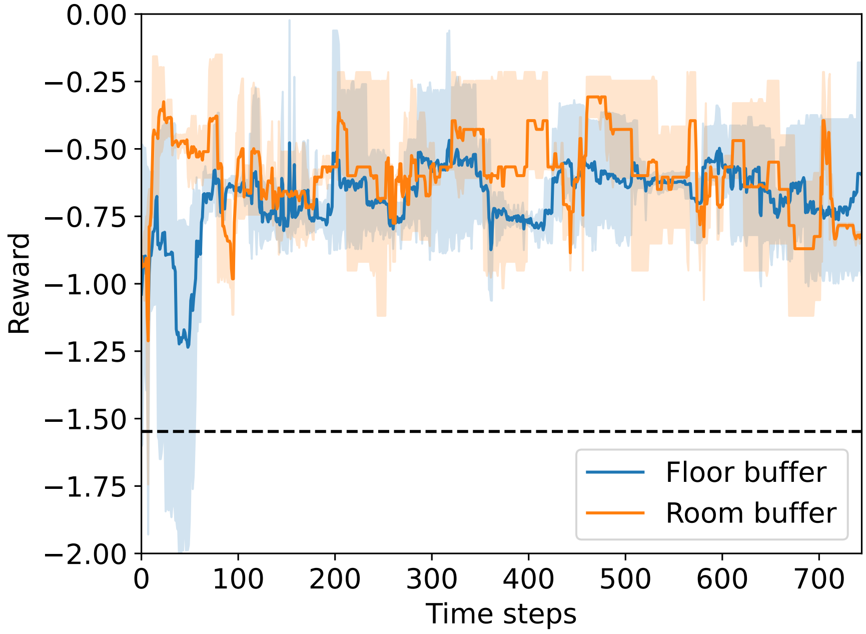


**Figure 2.13**: Room Batch vs Floor Batch

### 2.4.8 Generalization Experiments

**In-batch/Out-of-batch Experiment**

To examine the generalization of the BRL model, we test the learned policy on rooms where no data exist in the batch. Fig. 2.12 shows that out-of-batch (OOB) rooms cannot select proper actions to compensate for the OAT fluctuation during the week. The reward curves follow the OAT trend periodically, with clear peaks and valleys. This is reasonable since different zones might respond differently under the same VAV control action, due to the thermal dynamics in the HVAC and distance from VAV to zones.

**Room-specific/Floor-specific Experiment**

We validate if a room-specific policy is needed. For this, we use room-specific batch data as our expert policy and evaluate these same rooms. In Fig. 2.13 we observe that although both floor and room models yield consistent outcomes above the average. It is better to use a specific room buffer for a better fit of the room/zone thermal dynamics.

## 2.5 Conclusion and Future Works

Our simulator-free, multi-zone, BRL-based framework uses existing data as prior knowledge to learn the optimal policy without setting up complex, parameterized simulators. It saves energy compared with the default rule-based control method and maintains thermal comfort. To the best of our knowledge, our work is the first to improve and implement state-of-the-art BRL methods on real building HVAC control. We hope our research encourages domain experts to adopt BRL for real-world problems.

To further improve our control framework, we will update our building operation system to achieve a more frequent data writing rate. This way, we could train the model for the same number of time steps in a shorter time, hereby faster convergence of model. In addition, we will include rooms of different functionality, e.g. conference rooms, individual offices, and study areas, in our evaluation to create a more generalized model for HVAC control. Also, we would expand the action spaces by including chilling system control and economizers for more comprehensive optimization.

For methodology improvement, we plan to further investigate model-based method in offline mode, which uses dynamic models to generate a model buffer, and then the model buffer is also used to update the BRL model.

Chapter 2, in part, is a reprint of the material that appears in the proceeding of ACM/IEEE 13th International Conference on Cyber-Physical Systems (ICCPS 2022). By authors Hsin-Yu Liu, Bharathan Balaji, Sicun Gao, Rajesh Gupta, and Dezhi Hong with the title - "Safe HVAC control via batch reinforcement learning". The dissertation author is the primary investigator and author of this paper.

**Algorithm 2:** BCM training algorithm

```
Input  : Batch data $\mathcal{B}_f$ for a certain floor $f$, target network update rate $\tau$,
         mini-batch size $N$, max perturbation to selected actions $\Phi$, number of
         sampled actions $n$, minimum weighting $\lambda$, evaluation frequency
         $eval\_freq$, M-RL scaling parameter $\alpha_m$, and entropy temperature
         parameter $\tau_m$
Output: Updated target networks
Initialize: RL agent $BCM$, Q networks $Q_{\theta_1}, Q_{\theta_2}$, VAE generative network
  $G_\omega = \{E_{\omega_1}, D_{\omega_2}\}$, perturbation network $\xi_\phi$, random parameter $\omega, \phi, \theta_1, \theta_2$, and
  target networks $Q_{\theta_1'}, Q_{\theta_2'}, \xi_{\phi'}$ with $\theta_1^{'} \leftarrow \theta_1, \theta_2^{'} \leftarrow \theta_2, \phi^{'} \leftarrow \phi$
for $t \leftarrow 0$ to $T$ do
    Sample mini-batch $N$ transitions $(s, a, r, s')$ from $\mathcal{B}_f$;
    $\mu, \sigma = E_{\omega_1}(s, a), \tilde{a} = D_{\omega_2}(s, z), z \sim \mathcal{N}(\mu, \sigma)$
    $\omega \leftarrow \mathrm{argmin}_\omega$
    Sample $n$ actions: $\{a_i \sim G_\omega(s')\}_{i=1}^n$;
    Perturb each action: $\{a_i = a_i + \xi_\phi(s', a_i, \Phi)\}_{i=1}^n$;
    Set value target $y$ (Eqn.2.3);
    $\theta \leftarrow \mathrm{argmin}_\theta$;
    $\phi \leftarrow \mathrm{argmax}_\phi \, Q_{\theta_1}(s, a + \xi_\phi(s, a, \Phi)), a \sim G_\omega(s)$;
    Update target networks: $\theta_i^{'} \leftarrow \tau\theta + (1 - \tau)\theta_i^{'}$;
    $\phi^{'} \leftarrow \tau\phi + (1 - \tau)\phi^{'}$;
```

# Chapter 3

# Open-source Building HVAC Control Dataset for Batch Reinforcement Learning

## 3.1 Introduction

Reinforcement learning (RL) is widely studied in the building research area. Most studies focus on RL learning in an online paradigm [87, 126, 24, 38, 139, 146], assuming there is a simulation environment for RL models to interact with during training and evaluation stages before real-world deployment. Simulators such as EnergyPlus [18] and TRNSYS [53] are used to simulate the thermal states of a building. However, designing and calibrating such models for a large building is time-consuming and requires expertise.

In real-world scenarios, most large buildings are controlled via building management systems (BMS), where thermal data can be stored in the database. With advances in sensing technologies and machine learning, data-driven models have been more popular in recent research. Batch reinforcement learning, a data-driven approach that learns only from a fixed dataset generated with unknown behavioral policy, has not been explored widely in the building control community. BRL models are capable of learning the optimal policy without accurate environment models or simulation environments as oracles. In our study, we open-source both our dataset (https://github.com/HYDesmondLiu/B2RL) extracted from real building as well as the data generated with Sinergym [47], a building RL simulation environment which integrates EnegryPlus and BCVTB [131] with OpenAI Gym [13] interface. Furthermore, we experiment with several state-of-the-art BRL methods. The experimental results could be re-used as benchmarks for algorithm comparison.

## 3.2 Related Work

### 3.2.1 Building batch reinforcement learning

Previously, several studies implement fitted Q-iteration (FQI) and batch Q-learning [118, 92, 91, 135]. However, FQI and batch Q-learning, are based on pure off-policy algorithms. Fujimoto et al. [23] show that off-policy methods exacerbate the extrapolation error in a pure offline setting. These errors are attributed to Q-network training on historical data but exploratory actions yield policies that are different from the behavioral ones.

Recently, several studies related to building deep BRL research have emerged. Zhang et al. [140] apply CQL [59] on the CityLearn [120] testbed as simulator. We have outlined in the previous Chapter that we incorporate a Kullback-Leibler term in Q-update to penalize policies that are far from the previous one to improve from state-of-the-art BRL algorithm and deploy in real environments without setting up simulators.

### 3.2.2 Batch reinforcement learning datasets

As of this writing, D4RL [31] is the only open-source BRL dataset. The authors have generated various robotic control datasets. In our study, we open-source two building datasets, one contains real building buffers extracted from our building database with sensor readings, setpoints control history, and the estimated energy consumption calculated by Zonepac [6]. Then, we process them as Markov Decision Process (MDP) tuples. The other dataset is from a set of buffers that contain different qualities of transitions generated by pre-trained behavioral agents with simulation environments.

## 3.3 Approach and Results

### 3.3.1 Real building buffers

#### Data acquisition

The real building buffer is extracted from the readings of student labs in one of the school buildings as in the previous Chapter. The amount of data points in the buffers ranges from 170$\sim$260$K$, depending on the number of rooms involved and missing values. We obtain data of an entire year, from the beginning of July 2017 to the end of June 2018 for 15 rooms across 3 floors. The RL setup in our experiments is listed as below:

- *State*: Indoor air temperature, actual supply airflow, outside air temperature, and humidity.
- *Action*: Zone air temperature setpoint and actual supply airflow setpoint. Both are in continuous space and the action spaces are normalized in the range of $[-1, 1]$ as a standard RL setting.
- *Reward*: Our reward function is a linear combination of thermal comfort and energy consumption. The reward function at time step $t$ is:

$$R_t = -\alpha|TC_t| - \beta P_t, \tag{3.1}$$

  where $\alpha$, $\beta$ are the weights balancing different objectives and could be tuned to meet specific goals, $TC_t$ is the thermal comfort index at time $t$, $P_t$ is the HVAC power consumption at time $t$. We compute $P_t$ attributed to a thermal zone using heat transfer equations [6].

### BRL benchmarks

- Batch-constrained deep Q-learning (BCQ) [36]: As described in Chapter 2.
- Bootstrapping Error Accumulation Reduction (BEAR) [58]: BEAR identifies bootstrapping error as a key source of BRL instability. The algorithm mitigates out-of-distribution action selection by searching over the set of policies that is akin to the behavioral policy.

- Pessimistic Q-Learning (PQL) [70]: PQL uses pessimistic value estimates in the low-data regions in the Bellman optimality equation as well as the evaluation back-up. It can yield stronger guarantees when the concentrability assumption does not hold. PQL learns from policies that satisfy a bounded density ratio assumption similar to on-policy policy gradient methods.

**Experiment details**

Each algorithm is run in one room on each floor for an entire week so that outside air temperature (OAT) is the same. For instance, in one week we run algorithm A in rooms in the same stack on different floors, e.g. 2144, 3144, and 4144, and at the same time algorithm B runs on 2146, 3146, and 4146, and so forth. In each room, we train the algorithm for 1,000 time steps, which is about one week. We evaluate each algorithm in three different rooms (one room from each floor: 2F, 3F, and 4F). These rooms are of roughly the same size and occupancy capacity. Each time step is 10 minute due to the data writing rate in our BMS. More details of the experiments are described previously in our previous study [140].

Fig. 3.2 shows the learning curves of each algorithm, where each solid lines are the average reward of all runs for the same method; semi-transparent bands represent the range of all runs for a particular algorithm. And gray dotted vertical lines indicate 00:00AM of each day. The horizontal black dotted line is the average reward in the buffer. Fig. 3.3 shows the analysis of the optimization objectives in the reward function, for energy consumption, the default control method rule-based contorl (RBC) method is normalized to

1. For thermal comfort we show absolute averaged values.

As we need to calculate the thermal comfort level as required by our reward function, we adopt the widely used predicted mean vote (PMV) [27] measure as our thermal comfort index. In this metric, thermal comfort satisfaction ranges from $-3$ (cold) to 3 (hot), where PMV within the range of $-0.5$ to 0.5 is considered as thermal comfortable. We adopt the ASHRAE RP-884 thermal comfort data set [21] and train a simple gradient boosting tree (GBT) model [48] to predict the thermal comfort by taking the current thermal states given by our building system in real-time.

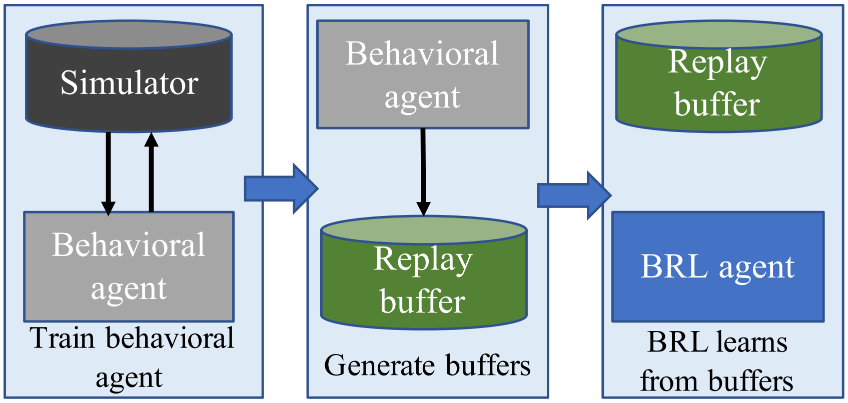


**Figure 3.1**: Flow of buffer generation and BRL training

### 3.3.2 Simulated buffers

#### Data acquisition

We adopt Sinergym, an open-source simulation and control framework for training RL agents [47]. It is compatible with EnergyPlus models using Python APIs. Our approach follows the BRL paradigm. (1) We first train behavioral RL agents for $500K$ timesteps

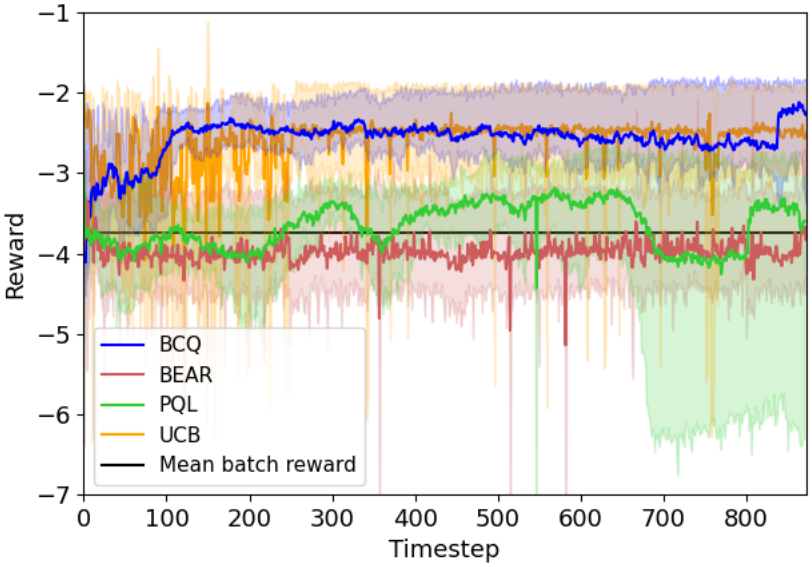


**Figure 3.2**: Episode reward comparison in real building

and select the one that gives the highest average score as the expert agent. We then run on a 5-zone building (See Appendix B.2), which is a single-floor building divided into 5 zones, 1 interior and 4 exteriors with 3 weather types: cool, hot, and mixed in continuous settings. We also experiment on two different kinds of response type, deterministic and stochastic. Then we generate an expert buffer with $500K$ transitions as the expert buffer. (2) A medium buffer is generated when the behavioral agent is trained ”halfway”, which means the evaluation score reaches half of the expert agents’ final average scores. (3) We randomly initialize the agent, which samples action from allowed action spaces with uniform distribution to generate buffers. (See Fig. 3.1)

- *State*: Site outdoor air dry bulb temperature, site outdoor air relative humidity, site wind speed, site wind direction, site diffuse solar radiation rate per area, site direct

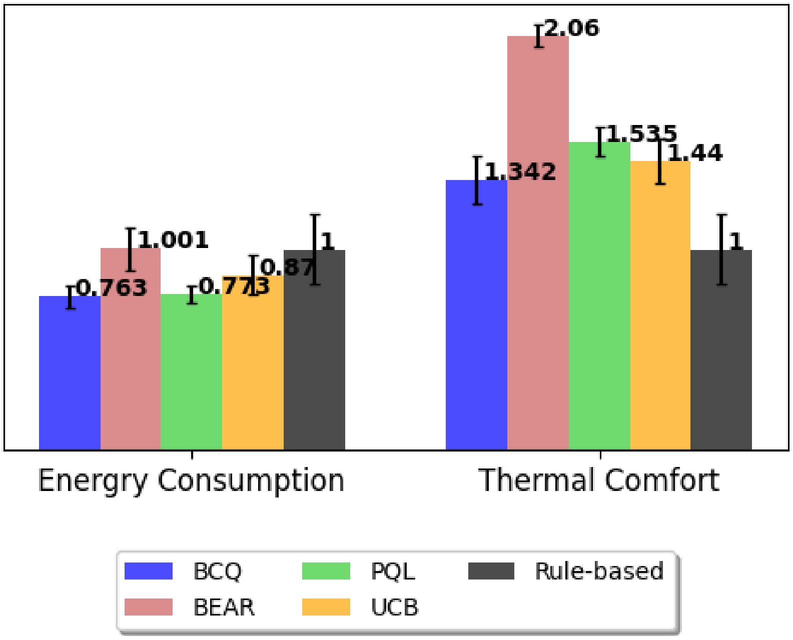


**Figure 3.3**: Optimization objectives analysis in real building

solar radiation rate per area, zone thermostat heating setpoint temperature, zone thermostat cooling setpoint temperature, zone air temperature, zone thermal comfort mean radiant temperature, zone air relative humidity, zone thermal comfort clothing value, zone thermal comfort Fanger model PPD, zone people occupant count, people air temperature, facility total HVAC electricity demand rate, current day, current month, and current hour.

- *Action*: Heating setpoint and cooling setpoint in continuous settings.

- *Reward*: We follow the default linear reward settings, which consider the energy consumption and the absolute difference in temperature comfort.

**BRL benchmarks**

With various qualities of buffers, we compare several most representative benchmarks in the BRL literature and summarize the average scores and standard deviation in the last 5 evaluations across 3 random seed runs (see Table 3.1). The scores of random policy is normalized to 0 and expert policy is normalized to 100.

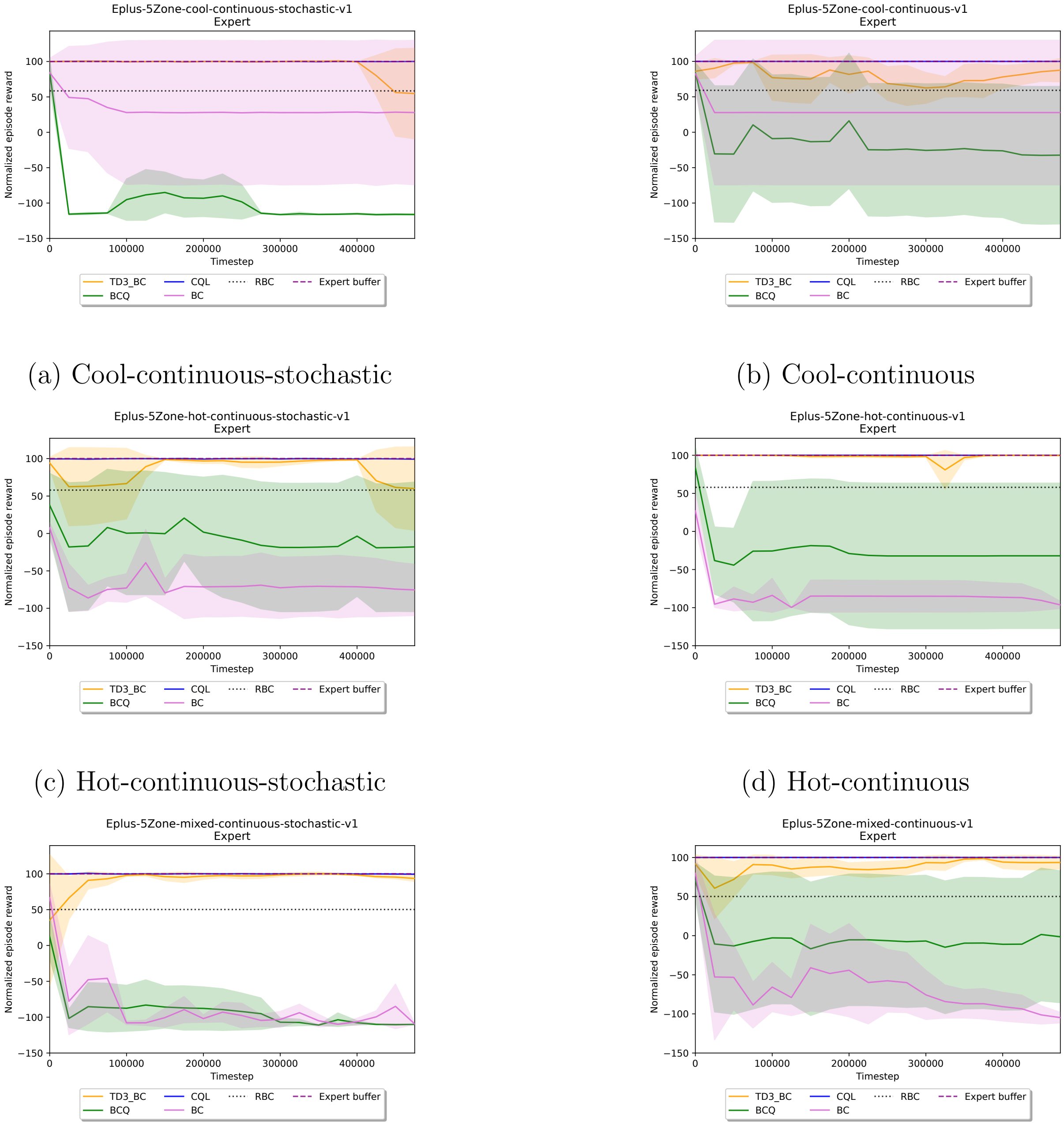
- TD3+BC: An offline version of TD3, it simply adds a behavior cloning term to regularize actor policy towards behavioral policy [33] combined with mini-batch Q-values and buffer states normalization for stability improvement.
- CQL: Conservative Q-learning [59], derived from SAC, learns a lower-bound estimates of the value function, by regularizing the Q-values during training.
- BC: Behavior cloning, we train a VAE to reconstruct action given state. It simply imitate the behavioral agent without reward signals.

We train each algorithm for $500K$ timesteps. For every $25K$ timesteps of training we evaluate the models for one episode. As an example, we illustrate BRL learning curves with expert buffers in Fig. 3.4.

## 3.4 Conclusion and Future Works

We open-source our building control datasets for both real buildings and simulation environments for BRL learning. The goal is to encourage building domain experts to explore opportunities in building-BRL research. We provide these datasets for researchers to implement fast prototyping without generating buffers on their own. Recently, many building-RL libraries are published [29, 96, 120] for the purpose of building RL training without the need to set up thermal simulators beforehand. Our future work is to generate more diverse buffers with various building environments and different weather types for BRL benchmarks.

Chapter 3, in part, is a reprint of the material that appears in the proceeding of the 9th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation (BuildSys' 2022). Bu authors Hsin-Yu Liu, Xiaohan Fu, Bharathan Balaji, Rajesh Gupta, and Dezhi Hong with the title - "B2RL: an open-source dataset for building batch reinforcement learning." The dissertation author is the primary investigator and author of this paper.



(a) Cool-continuous-stochastic

(b) Cool-continuous

(c) Hot-continuous-stochastic

(d) Hot-continuous

(e) Mixed-continuous-stochastic

(f) Mixed-continuous

**Figure 3.4**: Learning curves of BRL models that learn from expert buffers. Solid line shows the averaged value across three random seeds per algorithm, and the half-transparent region indicates the range with one standard deviation.

**Table 3.1**: Average normalized score over the final 5 evaluations and 3 random seeds.

| Environment | Buffer | TD3+BC | CQL | BCQ | BC |
|---|---|---|---|---|---|
| hot-deterministic | Expert | **99.72±0.1** | **100.00±0.00** | -32.02±0.07 | -89.2±3.95 |
| hot-deterministic | Medium | -49.59±8.19 | **67.65±17.06** | 13.41±16.59 | -12.55±7.27 |
| hot-deterministic | Random | -45.73±15.13 | -23.19±4.52 | **69.21±18.52** | -26.74±15.91 |
| mixed-deterministic | Expert | **94.67±2.04** | **100.00±0.00** | -6.22±5.24 | -95.46±6.6 |
| mixed-deterministic | Medium | 36.23±4.31 | 37.36±19.31 | **64.46±0.65** | -103.4±2.12 |
| mixed-deterministic | Random | -13.72±22.25 | -23.46±20.33 | -65.30±20.40 | -27.82±11.79 |
| cool-deterministic | Expert | 81.11±5.24 | **100.00±0.00** | -29.75±3.18 | 27.76±0 |
| cool-deterministic | Medium | -49.97±0.00 | 55.44±6.46 | **70.19±17.06** | 10.48±22.11 |
| cool-deterministic | Random | -58.40±3.21 | 12.99±2.28 | **27.77±31.39** | 8.62±41.97 |
| hot-stochastic | Expert | 77.69±17.18 | **99.49±0.20** | -15.35±5.92 | -72.86±1.73 |
| hot-stochastic | Medium | -14.85±0.00 | **39.93±2.64** | -62.21±19.31 | -10.45±12.85 |
| hot-stochastic | Random | -1.82±2.68 | **36.65±11.95** | -1.24±14.80 | 31.22±13.51 |
| mixed-stochastic | Expert | **96.61±2.13** | **99.77±0.26** | -108.38±2.58 | -102.02±9.32 |
| mixed-stochastic | Medium | 9.49±0.00 | **80.13±8.19** | 70.75±6.46 | -107.41±3.41 |
| mixed-stochastic | Random | 28.02±8.69 | **94.05±2.08** | -109.47±0.17 | 38.66±24.64 |
| cool-stochastic | Expert | 78.27±20.01 | **99.97±0.12** | -115.86±0.41 | 28.15±0.35 |
| cool-stochastic | Medium | 16.09±0.00 | **81.57±4.31** | -11.55±2.64 | -50.37±2.45 |
| cool-stochastic | Random | -44.33±16.01 | -97.35±2.09 | -53.92±10.07 | **25.44±13.42** |
| Sum | | 339.50±127.23 | **960.99±101.81** | -295.49±175.47 | -527.93±193.48 |

# Chapter 4

# Incorporating Existing Policies with Reinforcement Learning

## 4.1 Introduction

Buildings typically implement rule-based control, which adjusts the setpoints of actuators to co-optimize occupants' thermal comfort and energy efficiency. These rule-based control (RBC) systems codify the problem-solving know-how of human experts, akin to behavioral cloning policy learned from expert demonstration [41]. RBC is stable, robust, and without uncertainty, but lacks the flexibility to evolve over time.

Reinforcement learning (RL) can adapt to changes in the environment with a data-driven approach and improves the performance of HVAC systems control [129]. In online RL, the training of the control policy relies on a simulator that models the HVAC system. We use established building-RL simulation environments – *Sinergym* [47] for our experiments. However, when such a simulation model is not available, offline RL can be used to train a policy based on historical data [141]. We focus on improving upon existing RL algorithms for HVAC control where a rule-based policy already exists, which is a common scenario in real-world implementations. By combining the advantages of RL and rule-based methods, we aim to develop a stable and scalable algorithm without learning from scratch and utilize the existing knowledge in building a robust system.

In our work, we seek to answer the following research questions: ***How can we incorporate reinforcement learning models with an existing rule-based control policy to improve models' performance? Could this method be implemented in both online and offline settings as a unified approach?***

RL regularization methods are typically tailored to online or offline settings. For example, online methods encourage exploration to either improve estimations of non-greedy actions' values or to encourage the exploration to find an optimal policy [39]. On the other hand, offline methods favor exploitation since it is unlikely for offline models to accurately estimate uncharted state-action values with a static dataset [37, 133].

Our method builds on TD3+BC [33], a representative offline RL algorithm. TD3+BC makes minimal changes by adding a behavior cloning term to regularize the online TD3 [35] policy. In TD3+BC, the only policy to learn from is the behavioral policy that generates the buffer. Our dynamically-weighted algorithm regularizes RL policy using the better policy between an existing RBC policy and the behavioral policy. It can be incorporated into any existing actor-critic RL algorithms with minimal changes.

RUBICON considers RBC as a safe reference policy in which RL training can learn

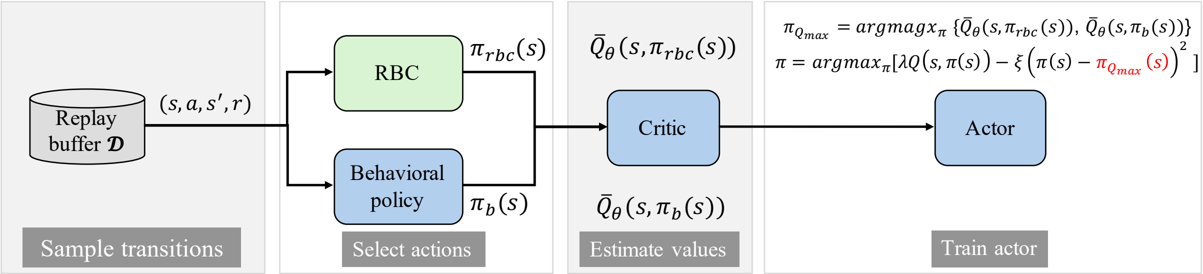

Replay buffer $\boldsymbol{\mathcal{D}}$
$(s, a, s', r)$
Sample transitions
RBC
$\pi_{rbc}(s)$
Behavioral policy
$\pi_b(s)$
Select actions
$\bar{Q}_\theta(s, \pi_{rbc}(s))$
Critic
$\bar{Q}_\theta(s, \pi_b(s))$
Estimate values
$\pi_{Q_{max}} = argmagx_\pi \{\bar{Q}_\theta(s, \pi_{rbc}(s)), \bar{Q}_\theta(s, \pi_b(s))\}$
$\pi = argmax_\pi[\lambda Q(s, \pi(s)) - \xi\left(\pi(s) - \pi_{Q_{max}}(s)\right)^2]$
Actor
Train actor

**Figure 4.1**: The flow of RUBICON: We incorporate the RBC policy and selectively update the actor with the policy (between RBC and behavioral) that has a higher estimated mean Q-value. It is a unified method for both online and offline approaches.

and improve. The actor selectively trains on either RBC or behavioral policy, depending on which policy yields a higher averaged Q-value in a mini-batch estimated by the critic network. The flow of RUBICON is shown in Fig. 4.1. Our proposed approach is distinct from prior work in the following aspects:

- We develop a unified regularization approach for both online and offline RL methods with minimal algorithmic modification.

- Rule-based control policy is directly incorporated into the policy update step to provide stability and robustness.

- We introduce a dynamic weighting method in actor-critic settings. The actor loss is varied from time step to time step depending on the average Q-value estimate of behavioral policy and RBC policy predicted from the value networks.

To our knowledge, previously RBC is only used as hard constraints or heuristics in RL settings, and we are the first to incorporate an existing RBC policy directly into actor-critic algorithms.

## 4.2 Related Work

**Building RL control** Prior research has demonstrated that building RL control policy could outperform RBC in both online and offline settings. Researchers have studied extensively for HVAC control with online RL methods [40, 126, 136]. [145] developed a framework for whole building HVAC (heating, ventilation, air-conditioning) control in online settings to achieve a 16.7% heating demand reduction cf. RBC control. OCTOPUS holistically controls subsystems in modern buildings to get a 14.26% energy saving cf. RBC policy [23]. [135] implemented an RL control for LowEx building systems with a 11.47% improvement on cumulative net power output than RBC.

With offline RL, [141] applied a state-of-the-art method and demonstrated a $12 \sim 35\%$ of reduction in ramping. [68] incorporated a Kullback-Leibler (KL) divergence constraint during the training of an offline RL agent to penalize policies that are far away from the previous updates for stability, and achieve a 16.7% of energy reduction cf. the default RBC control.

**RL + RBC** The combination of RL and RBC has been explored in many studies, where RBCs are primarily used as auxiliary constraints or guiding mechanism. [61] propose to use two modules in their control flow, one for continuous control with RL agent and a discrete one controlled by RBC. [125] improve RL with low-level rule-based trajectory modification to achieve a safe and efficient lane-change behavior. [150] incorporate

RBC for generating the closed-loop trajectory and reducing the exploration space for RL pre-processing. [11] use a learning process to fine-tune the performance of a rule-based controller. [86] first train RL proximal policy optimization (PPO) [100] agents to master matching some of the problem rules and constraints, then RL is used to inject experiences to guide various evolutionary/stochastic algorithms. [65] learn RBC parameters via RL methods. These previous methods incorporate RBC in the flow as heuristics or as hard constraints. Instead, we directly incorporate RBC policy in RL training in an algorithmic way.

**Online RL regularization** We use state-of-the-art TD3 to compare the online baselines. It applies target policy smoothing regularization to avoid overfitting in the value estimate with deterministic policies. TRPO [99] uses a trust region constraint based on KL-divergence between old and new policy distributions for robust policy updates. SAC [39] uses soft policy iteration for learning optimal maximum entropy policies. Munchausen-RL [123] regularizes policy updates with a KL-divergence penalty similar to TRPO, and adds a scaled entropy term to penalize policy that is far from uniform policy.

**Offline RL regularization** Offline RL is more conservative compared with online methods as it depends only on the logged interactions generated by unknown policies. It suffers from extrapolation errors induced by selecting out-of-distribution actions. Since offline RL policies are learned entirely from a static dataset, it is unlikely for value networks to accurately estimate values when there is no sufficient state-action visitation. Thus, regularization methods become more prominent in offline settings. Batch-constrained deep Q-learning (BCQ) [37], one of the pioneers of offline RL, ascribes extrapolation errors to three main factors: absent data, model bias, and training mismatch. It mitigates the errors by deploying a variational autoencoder (VAE) to reconstruct the action given a state using the data collected by the behavioral policy. Then regularize the divergence between the learned policy and the behavioral policy. The offline baseline method we compare to in our study is TD3+BC. It starts from the online method TD3, and adds a behavior cloning term in the policy update to regularize the actor to imitate the behavioral policy and avoid selecting out-of-distribution actions. BRAC [133] studies both value penalty and policy regularization with multiple divergence metrics (KL, maximum mean discrepancy (MMD), and Wasserstein) to regularize the actor's policy based on the behavioral policy. FisherBRC [54] incorporates a gradient penalty regularizer for the state-action value offset term and demonstrates the equivalence to Fisher divergence regularization. CQL [59] learns a conservative, lower-bound estimate in the value network via regularizing Q-values. Model-based method, e.g. COMBO [137] regularizes the value function on out-of-support transitions generated via environment dynamic models' rollouts.

**Conservative RL** [114] use a priori unknown safety constraint that depends on state-action and satisfies certain regularity conditions with a Gaussian prior. [4] propose to synthesize a reactive system called a *shield* to monitor the actions and correct them if violations are caused. [108] use a pre-specified "safety" threshold as a requirement and express it via a Gaussian process prior.

All of these prior works use data collected by a behavioral policy and do not assume access to any existing policy. The behavioral policy used in experiments is typically an unknown or partially trained agent. In contrast, we assume direct access to a robust behavioral policy in the form of rule-based control. While this assumption may not hold for other applications where there might not be pre-existing policies, rule-based control policies are routinely deployed in industrial control settings, such as building HVAC control.

We incorporate a robust reference policy derived by human experts to improve RL policy. The rule-based control policy reduces uncertainty due to its deterministic behavior. On the opposite, the deep learning model is affected by random initialization conditions, even if trained on the same dataset, as varied initialization conditions might lead to different policies. RUBICON demonstrates a substantial reduction of standard deviations between different randomly initialized conditions across varied tasks.

## 4.3 Terminologies and Problem Formulation

In reinforcement learning, an agent interacts with the environment and sequentially selects actions based on its policy at every time step. The problem can be formulated as a Markov Decision Process (MDP) defined by a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{R}, p, \gamma)$, with state space $\mathcal{S}$, action space $\mathcal{A}$, reward function $\mathcal{R}$, transition dynamics $p$, and discount factor $\gamma \in [0, 1)$. The goal is to maximize the expectation of the cumulative discounted rewards, denoted by $R_t = \sum_{i=t+1}^{\infty} \gamma^i r(s_i, a_i, s_{i+1})$ [109]. The agent's behavior is determined by a policy $\pi : \mathcal{S} \rightarrow \mathcal{A}$, which maps states to actions either in a deterministic approach or with a probability distribution. The expected return following the policy from a given state $s$ is the action-value function $Q^{\pi}(s, a) = \mathbb{E}_{\pi}[\sum_{t=0}^{\infty} \gamma^t R_{t+1} | s_0 = s, a_0 = a]$ by taking action $a$.

We conduct our experiments with the building RL environments [47]. The objective of the agent is to maintain a comfortable thermal environment with minimal energy use. The state consists of indoor/outdoor temperatures, time/day, occupant count, thermal comfort, and related sensor data. The action adjusts the temperature setpoint of the thermostat. The reward is a linear combination of occupants' thermal comfort and energy consumption. The environment is a single-floor building divided into 5 zones, with 1 interior and 4 exterior rooms.

The details about the RL settings in our problem are described below:

- **State**: Site outdoor air dry bulb temperature, site outdoor air relative humidity, site wind speed, site wind direction, site diffuse solar radiation rate per area, site direct solar radiation rate per area, zone thermostat heating setpoint temperature, zone thermostat cooling setpoint temperature, zone air temperature, zone thermal comfort mean radiant temperature, zone air relative humidity, zone thermal comfort clothing value, zone thermal comfort Fanger model PPD (predicted percentage of dissatisfied), zone people occupant count, people air temperature, facility total HVAC electricity demand rate, current day, current month, and current hour.

- **Action**: Heating setpoint and cooling setpoint in continuous settings for the interior zones.

- **Reward**: We follow the default linear reward setting, which considers the energy consumption and the absolute difference to temperature comfort.

- **Environment**: A single floor building with an area of $463.6m^2$ divided into 5 zones, 1 interior, and 4 exteriors. The HVAC system is a packaged VAV (variable air volume) (DX (direct expansion) cooling coil and gas heating coils) with fully auto-sized input. And the simulation period of one episode is a full year. The weather types are classified according to the U.S. Department of Energy (DOE) standard [79]. The weather type details and their representative geometric locations are listed below based on TMY3 datasets [60]:

- **Cool marine**: Washington, USA. The mean annual temperature and mean annual relative humidity are 9.3°C and 81.1% respectively.
- **Hot dry**: Arizona, USA with mean annual temperature of 21.7°C and a mean annual relative humidity of 34.9%
- **Mixed humid**: New York, USA with a mean annual temperature of 12.6°C and a mean annual relative humidity of 68.5%

## 4.4 Rule-based incorporated control regularization

Our goal is to improve an agent's ability to learn with the assistance of human experts' domain knowledge in both online and offline settings. In certain problems, we could configure accurate simulators as oracles so we can safely learn with online RL methods or there might be existing simulators. For example, in robotic control [112], Go [106], and video games [76]. However, for most real-world problems, it is unlikely or it is time-consuming and requires a domain expertise to build a functional simulator for each environment (e.g. building thermal simulators), or it can be dangerous or risky to evaluate partially trained policy directly in real environments(e.g. healthcare and financial trading). Offline RL al-

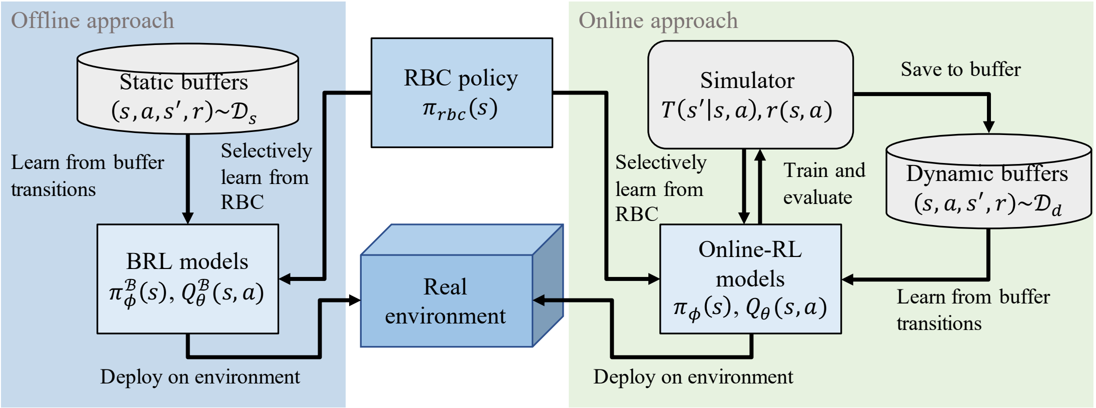

Offline approach
Static buffers
$(s,a,s',r)\sim\mathcal{D}_s$
Learn from buffer transitions
Selectively learn from RBC
BRL models
$\pi^{\mathcal{B}}_\phi(s), Q^{\mathcal{B}}_\theta(s,a)$
Deploy on environment
RBC policy
$\pi_{rbc}(s)$
Real environment
Online approach
Simulator
$T(s'|s,a), r(s,a)$
Save to buffer
Dynamic buffers
$(s,a,s',r)\sim\mathcal{D}_d$
Selectively learn from RBC
Train and evaluate
Online-RL models
$\pi_\phi(s), Q_\theta(s,a)$
Learn from buffer transitions
Deploy on environment

**Figure 4.2**: Our proposed method, RUBICON, incorporates RBC into RL to improve stability in building HVAC control. It could be applied to both online and offline approaches.

gorithms, on the other hand, rely on historical data collected by an existing but unknown behavioral policy. The objective is to learn a policy that improves on the behavioral policy measured by episodic rewards. In Fig. 4.2, we illustrate how RUBICON accommodates both the online and offline training paradigms.

Our algorithm builds on existing actor-critic algorithms TD3 and TD3+BC. We only modify the policy update with the incorporated rule-based control policy selectively and use the critic as-is. Therefore, we focus our discussion on the policy update of the algorithm. TD3 is derived from DDPG [105], it mitigates the function approximation error with double Q-learning and delayed policy updates. TD3+BC is an offline RL algorithm adapted from TD3, and is one of the state-of-the-art offline RL methods evaluated with D4RL datasets [31]. TD3+BC adds a behavior cloning term to the policy update step to penalize the policy that is far away from the behavioral policy (Eq. 4.1).

$$\pi = \underset{\pi}{\operatorname{argmax}} \, \mathbb{E}_{(s,a)\sim\mathcal{D}} \left[ \lambda Q(s, \pi(s)) - (\pi(s) - a)^2 \right] \tag{4.1}$$

$$\lambda = \frac{\alpha}{\frac{1}{N}\sum_{(s_i,a_i)} |Q(s_i, a_i)|} \tag{4.2}$$

In Eq. 4.1, $\lambda$ is decided by the averaged mini-batch Q-estimate and a hyperaparameter $\alpha$ to adjust between RL and imitation learning (Eq. 4.2), where $N$ is the size of the batch.

Our method, RUBICON, dynamically weighs both TD3 and

TD3+BC's policy update steps with either RBC policy or behavioral policy in each training iteration. In Eq. 4.3, we replace the actions $a$ sampled from the buffers in Eq. 4.1 with $\pi_{Q_{max}}(s)$ and add a hyperparameter $\xi$ to integrate TD3 and TD3+BC methods as one. We replace the notation of sampled actions $a$ in TD3+BC with behavioral policy $\pi_b(s)$ to avoid confusion. Details of the hyperparameter settings in our work are in Appendix B.3.

$$\pi = \underset{\pi}{\operatorname{argmax}} \, \mathbb{E}_{s\sim\mathcal{D}} \left[ \lambda Q(s, \pi(s)) - \xi(\pi(s) - \pi_{Q_{max}}(s))^2 \right] \tag{4.3}$$

$$\pi_{Q_{max}}(s) = \underset{\pi}{\operatorname{argmax}} \{ \bar{Q}(s, \pi_b(s)), \bar{Q}(s, \pi_{rbc}(s)) \} \tag{4.4}$$

Every time when the policy is being updated, given the states $s$ of the sampled mini-batch, the behavioral policy $\pi_b(s)$ and the RBC policy $\pi_{rbc}(s)$ select actions in a deterministic fashion. The state-action pairs' Q-values are estimated by the critic, the average of the Q-value estimations in the mini-batches are $\bar{Q}(s, \pi_b(s))$ and $\bar{Q}(s, \pi_{rbc}(s))$. The actor models dynamically choose the selected actions decided by the policy with a higher average

Q-value to be regularized from in each policy update step, i.e. the actor loss function is dynamically weighted. By describing it as dynamically weighted, it means that the actor loss is changing from one iteration to another since it is automatically decided by Eq. 4.4. In offline settings the actor loss is either regularized with the behavioral policy or the RBC policy; in online settings, it is either regularized with the RBC policy or learning as is without behavioral cloning term. In online settings, the behavioral policies are the older versions of the policy used to generate the transitions in the buffer, and in offline settings, it is an unknown policy.

The reason we choose the average as the metric to decide which set of transitions to learn from instead of selecting each transition with higher estimated value (each batch is a combination of $\pi_b(s)$ and $\pi_{rbc}(s)$ ) is that if we choose by each transition we will lose the information on which state-action visitations lead to worse values, the model will then suffer from the imbalanced data problem. The credit-blame assignment is essential in RL learning convergence and the experience replay can help speed up the propagation process [67]. Furthermore, the over-estimation of Q-values would be more severe. The algorithm of our method is given in Alg. 3. Where $d$ is the policy update frequency, the noise $\epsilon$ added to the policy is sampled with Gaussian $\mathcal{N}(0, \sigma)$ and clipped by $c$. In both online and offline approaches, the policy update follows Eq. 4.3 and 4.4 with different hyperparameter settings.

Our rule-based control algorithm is derived from the rule-based controller in Sinergym's [47] example. For the purpose of computation efficiency and to fit the batch settings in our algorithm, we vectorize the original RBC policy. The rules are simple and intuitive, and could generalize well: First, we get the datetime information we need from the states. Then, we get the seasonal comfort temperature zone for every transition. If the indoor air temperature (IAT) is below the lower bound of the comfort zone, then we set both cooling and heating setpoints a degree higher (measured in Celcius degrees). On the opposite, if the IAT is above the upper bound of the comfort zone, then we set both the heating and cooling setpoints a degree lower than the current setpoints. Finally, we examine if the current datetime is in the office hours. If not, then the setpoints are set to be $(18.33, 23.33)$ (°C) for the purpose of energy reduction since occupants' thermal comfort is not important in these time periods assuming zero occupancy.

## 4.5 Experiments

In our experiments, we use two environment response types: deterministic and stochastic. A Gaussian noise with $\mu$=0 and $\sigma$=2.5 is added to the outside temperature from episode to episode in the stochastic environments. We also consider three weather types: hot, cool, and mixed. In the results of all the tables and figures, "*hot-deterministic*" indicates that the task is learned and evaluated with the hot weather condition and deterministic environment. Similarly, we have all six combinations such as "*cool-stochastic*", etc. All scores in this paper are normalized with expert policy as 100 and random policy as 0.

**Algorithm 3:** RUBICON

```
Initialize critic networks $Q_{\theta_1}, Q_{\theta_2}$ ,actor network $\pi_\phi$, with random parameters $\theta_1$, $\theta_2$, $\phi$,
  target networks $\theta'_1 \leftarrow \theta_1$, $\theta'_2 \leftarrow \theta_2$, $\phi' \leftarrow \phi$, RBC policy $\pi_{rbc}$, and replay buffer or load
  buffer $\mathcal{B}$
for $t = 1$ to $T$ do
    if online then
        Select action with exploration noise
        $a \sim \pi_\phi(s) + \epsilon$, $\epsilon \sim \mathcal{N}(0, \sigma)$
        Observe reward $r$ and next state $s'$
        Store transition $(s, a, r, s')$ in $\mathcal{B}$
    Sample mini-batch of $N$ transitions $(s, a, r, s')$ from $\mathcal{B}$
    $\tilde{a} \leftarrow \pi_{\phi'}(s') + \epsilon$, $\epsilon \sim clip(\mathcal{N}(0, \tilde{\sigma}), -c, c)$
    $y \leftarrow r + \gamma \min_{j=1,2} Q_{\theta'_j}(s', \tilde{a})$
    Update critics $\theta_j \leftarrow \text{argmin}_{\theta_j} N^{-1} \sum (y - Q_{\theta_j}(s, a))^2$
    if $t$ mod $d$ then
        Update $\phi$ by policy gradient:
        $\nabla_\phi J(\phi) = N^{-1} \sum \nabla_a Q_{\theta_1}(s, a)|_{a=\pi_\phi(s)} \nabla_\phi \pi_\phi(s)$
        Policy update follows Eq. 4.3 and 4.4
        Calculate $\nabla_\phi J(\phi)$
        Update target networks:
        $\theta'_j \leftarrow \tau\theta_j + (1 - \tau)\theta'_j$
        $\phi' \leftarrow \tau\phi + (1 - \tau)\phi'$
```

**Algorithm 4:** Rule-based control policy

```
Input  : Current datetime $current\_dt$, indoor air temperature $IAT$, zone thermostat
         heating setpoint temperature $a_h$, and zone thermostat cooling setpoint
         temperature $a_c$, obtained from the states with size $N$
Output: Actions selected by RBC
for $i$ in $N$ do
    $season\_comfort\_zone_i = get\_season\_comfort(current\_dt_i)$
    if $IAT_i >= max(season\_comfort\_zone_i)$ then
        $a_{h_i} = a_{h_i} - 1$
        $a_{c_i} = a_{c_i} - 1$
    if $IAT_i < min(season\_comfort\_zone_i)$ then
        $a_{h_i} = a_{h_i} + 1$
        $a_{c_i} = a_{c_i} + 1$
    $a_i = (a_{h_i}, a_{c_i})$
    if $current\_dt_i.weekday \geq 5$ or $current\_dt_i.hour$ in range(22,6) then
        $a_i = (18.33, 23.33)$(°C)
```

### 4.5.1 Offline approach

#### Benchmarking Experiment

First, we consider the offline approach, where no simulator exists but historical data is available. We follow the standard procedure for BRL evaluation [31]:

(1.) Train behavioral agents for 500K time steps, then compare the most representative algorithms DDPG, TD3, and SAC (learning curves are shown in Fig. B.4). The online methods we compare are described below:

- **DDPG**: Deep deterministic policy gradient is a method that combines the actor-critic approach and deep Q-network (DQN) [76]. It is capable of dealing with continuous action space problems via policy gradient in a deterministic approach which outperforms the stochastic policy methods in high-dimensional tasks.

- **SAC**: Soft actor-critic, an off-policy maximum entropy RL algorithm that encourages exploration. They empirically show that SAC yields a better sample efficiency than DDPG.

- **TD3**: Twin delayed deep deterministic policy gradient algorithm, it reduces overestimation with double Q-learning, combines with target networks to limit errors from imprecise function approximation.

(2.) Select the best agent as our expert agent and generate buffers with it for 500K time steps. A medium agent is trained "halfway", which means that an agent is trained most closely to an agent with the evaluation performance half the performance as the expert agent. And a random agent which samples actions randomly and generates buffers.

(3.) Train BRL models for 500K time steps and evaluate the policy every 25K time steps in all buffers mentioned above in step (2.). We show the detailed learning curves in Appendix B.2. Normalized and averaged scores across runs are shown in Table 4.1. The offline methods we compare with are listed below:

- **TD3+BC**: An offline version of TD3, it adds a behavior cloning term to regularize policy towards behavioral policy combined with mini-batch Q-values and buffer states normalization for stability improvement.

- **CQL**: Conservative Q-learning, derived from SAC, it learns a lower-bound estimate of the value function by regularizing the Q-values during training.
- **BCQ**: Batch-constrained deep Q-learning, it implements a variational autoencoder (VAE) [52] to reconstruct the action given the state. And adds perturbation in actor on the policy, the degree of perturbation and size of mini-batch can be adjusted in order to behave more like a traditional RL method or imitation learning.
- **BC**: Behavior cloning, we train a VAE to reconstruct action given state. It simply imitates the behavioral agent without reward signals.

In Table 4.1[1], we observe that RUBICON outperforms all other benchmarks in overall score across weather types, random seeds, and environment types. To breakdown the reward scores into the optimization objectives, RUBICON achieves an overall 10.11% of energy reduction and 34.44% in comfort penalty cf. to the state-of-the-art method CQL. Other BRL methods show good performance either in specific tasks or with a specific randomly initialized configuration; however, overall they are more unstable cf. RUBICON. Our method provides more robust and more consistent performance across all variants and demonstrates the ability to generalize across various weather types and response modes of tasks. Also, as we can see in Fig. 4.3, learning from both medium buffer and RBC policy, RUBICON improves on both their best performances. The standard deviation of RUBICON's scores is the least among the policies evaluated, which means our policy is the

[1] Average normalized score over the final 5 evaluations and 3 random seeds. Values followed by ± correspond to the standard deviation over the last 5 evaluations across runs. Some scores with a standard deviation of 0 is caused by the round down of normalized scores, they are negligible numbers. But not exact zeros.

**Table 4.1**: BRL methods benchmark

| Environment | Buffer | RUBICON | TD3+BC | CQL | BCQ | BC |
|---|---|---|---|---|---|---|
| hot-deterministic | Expert | 86.13±17.83 | 99.72±0.42 | **100±0** | -32.01±95.46 | -89.2±14.84 |
| hot-deterministic | Medium | 64.91±18.02 | -49.58±13.52 | **67.64±32.83** | 13.4±51.24 | -26.74±26.47 |
| hot-deterministic | Random | 62.7±14.36 | -45.73±44.8 | -23.19±76.76 | **69.2±33.61** | -12.55±74.63 |
| mixed-deterministic | Expert | 81±25.94 | 94.66±7.36 | **100±0** | -6.22±84.27 | -95.46±14.78 |
| mixed-deterministic | Medium | **86.84±12.39** | 36.23±56.33 | 37.36±86.8 | 64.45±37.4 | -27.82±63.16 |
| mixed-deterministic | Random | **68.83±4.93** | -13.71±57.06 | -23.46±83.61 | -65.29±48.84 | -103.4±7.45 |
| cool-deterministic | Expert | 98±2.78 | 81.11±16.88 | **100±0** | -29.74±95.89 | 27.76±102.15 |
| cool-deterministic | Medium | **72.2±8.07** | -49.97±36.4 | 55.44±49 | 70.18±14.42 | 8.62±45.32 |
| cool-deterministic | Random | **66.5±0** | -58.4±19.25 | 12.98±73.04 | 27.77±63.67 | 10.48±70.79 |
| hot-stochastic | Expert | 99.01±0.56 | 77.69±30.48 | **99.49±0.24** | -15.34±84.51 | -72.86±38.25 |
| hot-stochastic | Medium | **59.72±5.29** | -14.84±66.04 | 39.92±56.67 | -62.2±5.25 | 31.22±66.26 |
| hot-stochastic | Random | **68.83±21.26** | -1.82±73.31 | 36.64±67.61 | -1.23±68.86 | -10.45±61.91 |
| mixed-stochastic | Expert | 94.16±8.12 | 96.6±2.14 | **99.77±0.21** | -108.38±2.83 | -102.02±9.26 |
| mixed-stochastic | Medium | **87.23±12.34** | 9.48±81.06 | 80.13±20.78 | 70.75±9.9 | 38.66±48.02 |
| mixed-stochastic | Random | 67.03±6.26 | 28.01±72.79 | **94.04±5.87** | -109.46±0.77 | -107.41±4.36 |
| cool-stochastic | Expert | 53.58±65.53 | 78.27±31.08 | **99.97±0.32** | -115.85±0.98 | 28.15±101.52 |
| cool-stochastic | Medium | 68.07±0.46 | 16.09±69.41 | **81.56±18.01** | -11.55±56.13 | 25.44±35.57 |
| cool-stochastic | Random | **67.55±1.14** | -44.33±36.36 | -97.35±11.07 | -53.92±78.06 | -50.37±83.99 |
| Sum |  | **1352.37±225.38** | 339.49±714.77 | 960.98±582.88 | -295.48±832.17 | -527.93±868.81 |

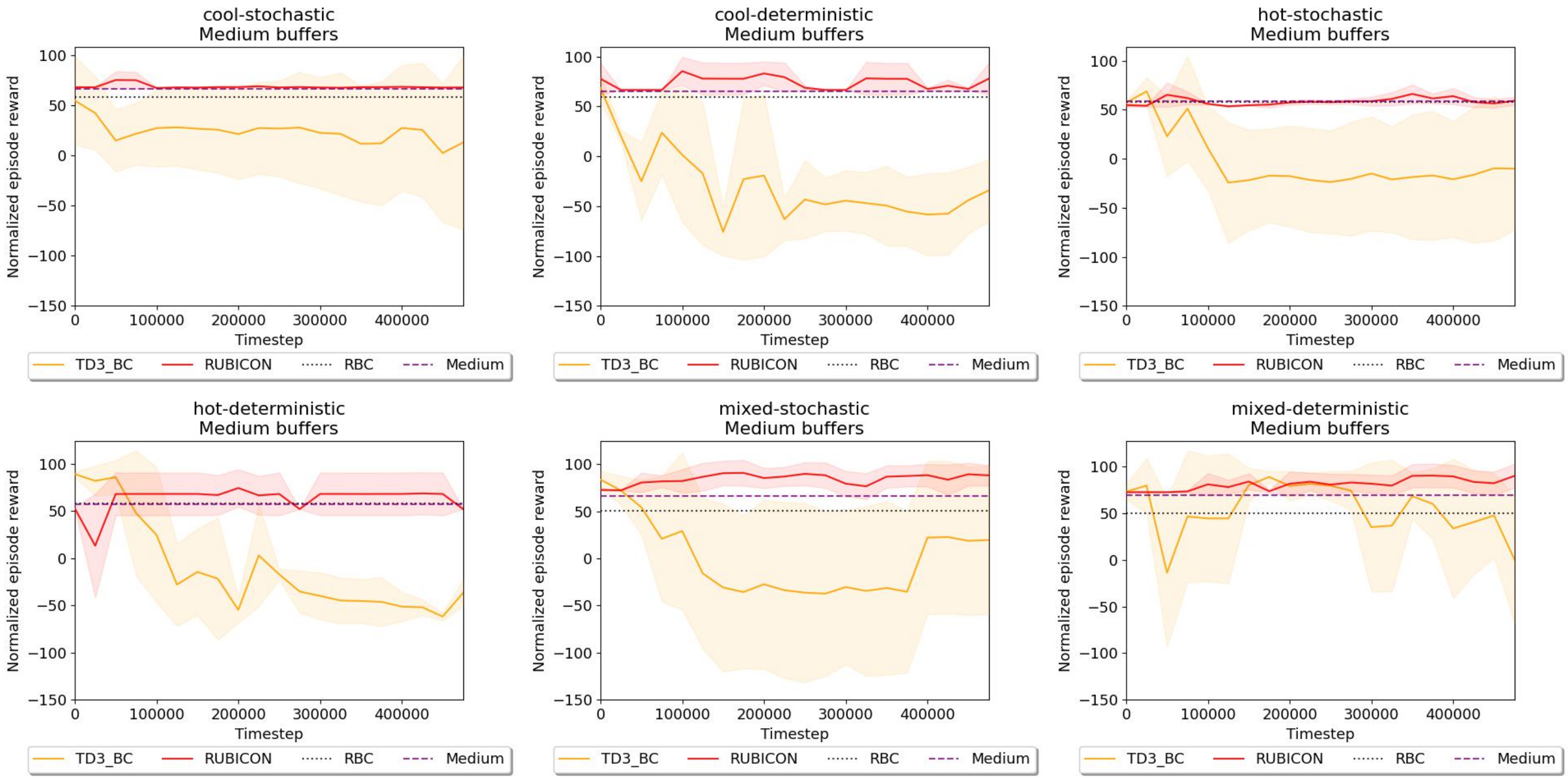


**Figure 4.3**: Learning curves of RUBICON and the baseline method TD3+BC with medium buffers. All learning curves are plotted with solid lines indicating averaged values and the half-transparent region is one standard deviation.

most stable one cf. others. We include the BRL learning curves with expert and random buffers in Appendix B.2

In the following sections, we conducted several robustness and ablation experiments to demonstrate the necessity of our enhancements.

### Transfer experiment

In a real-world scenario where we might have existing building control data in one building, but no data for a new building with different weather data distribution. Given existing buffers and we want to use them as prior knowledge to combine with RBC policy and transfer the model to another weather type where we have no data. We experiment with the medium buffers in stochastic environments. The results shown in Table B.2 and Fig. 4.4 indicate that our method is capable of transferring from one weather condition to another with comparable performance without any hyperparameter or RBC policy change. As the results demonstrate, due to the diversity of the mixed weather, RUBICON improves learning in cool and hot weather. On the other hand, transfer from monotonic weather conditions leads to worse returns.

### Ablation Experiments

**RBC buffer experiments** To differentiate RUBICON from directly learning from RBC buffers, we conduct the experiment of BRL models learning from the RBC buffers, i.e. learns from a buffer generated by RBC policy. The learning curves are illustrated in Figure 4.5. The results in Table B.3 indicate that even when learning with RBC buffers

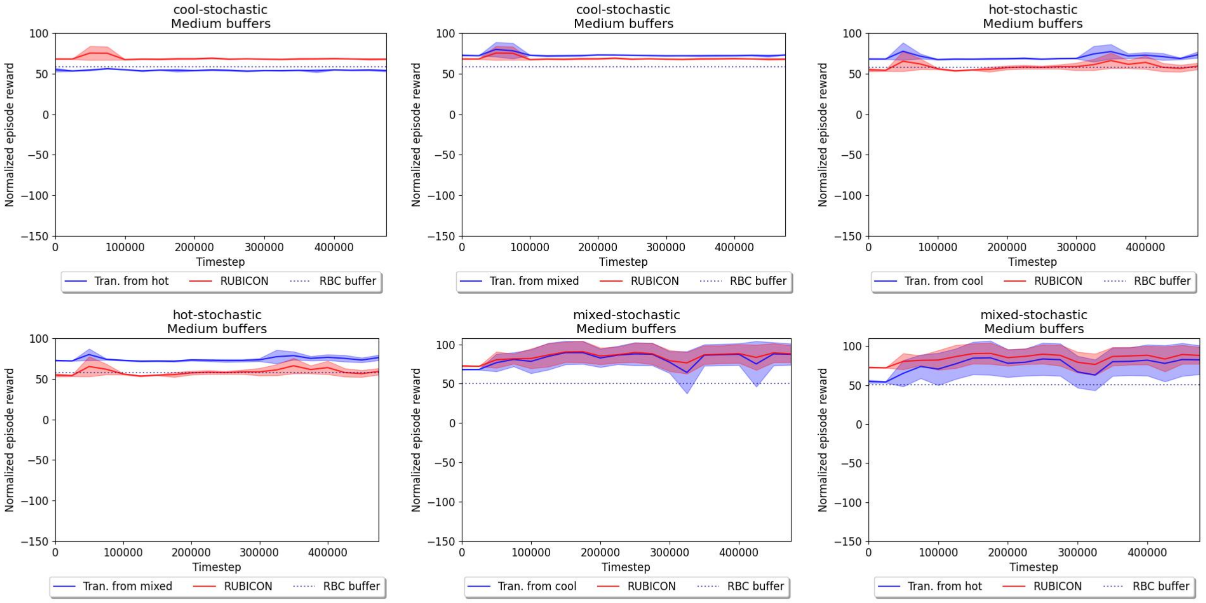


**Figure 4.4**: Learning curves of BRL models transferred from other weather types

with RBC policy itself, RUBICON could still outperform RBC policy due to its learning ability. However, due to the monotony of RBC policy, the improvement is limited. It shows the importance of incorporating RBC policy and RL policy.

**Mixed buffer experiments** In order to evaluate if mixing the buffers (of RBC buffer and the original buffer to learn from) is equivalent to RUBICON, we conduct experiments by mixing 50% of transitions in RBC buffer with 50% of transition in the original buffer to learn from. The result is shown in Figure 4.6 and Table B.6. It indicates our selective algorithm is necessary to dynamically decide if RBC policy or the behavioral policy to learn from instead of randomly trained on both.

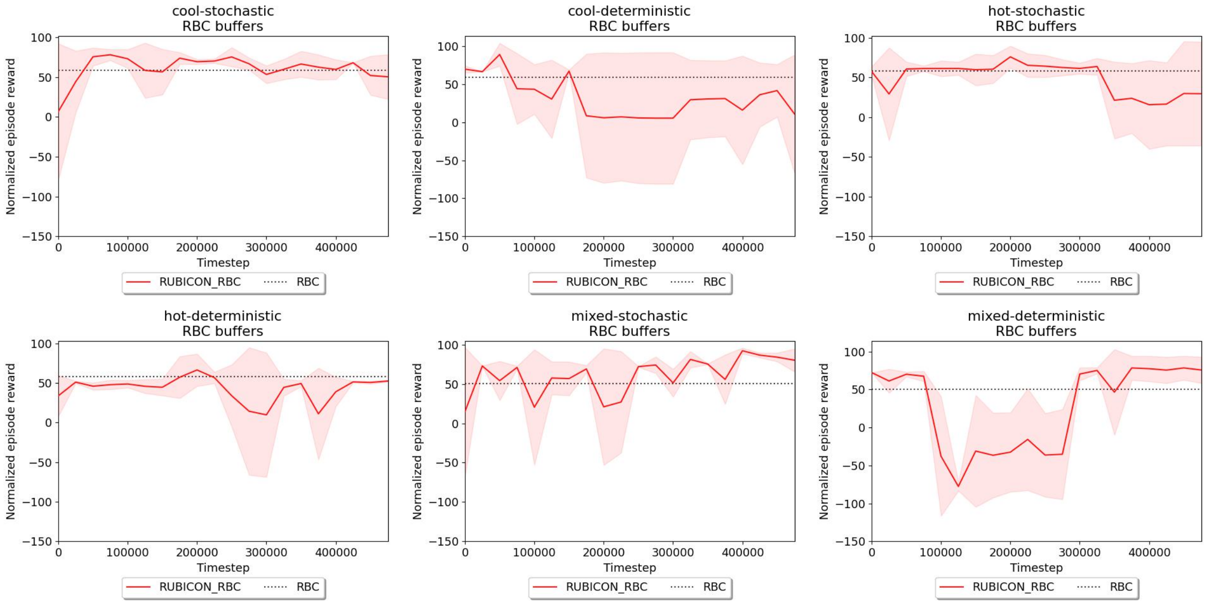


**Figure 4.5**: Learning curves of RUBICON learns from RBC buffers

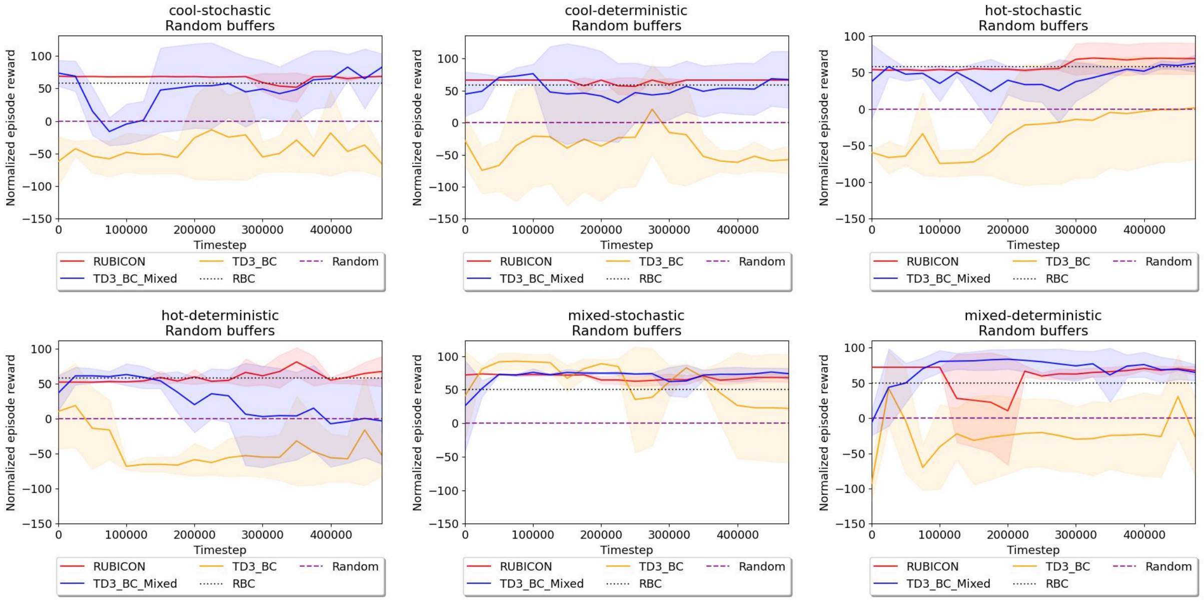


**Figure 4.6**: Learning curves comparing RUBICON and TD3+BC to TD3+BC learns from a mixture of 50% amount of transitions from the random buffer and 50% amount of transitions from the RBC buffer in stochastic environments

### Robustness Experiments

**Data efficiency experiment** We conduct the experiments with buffers of only one year of data ($35,040$ transitions). Data efficiency is a challenge for model-free RL to yield accurate value estimation as it is considered data inefficient generally. This experiment is designed to observe how RUBICON adapts in a scenario where there is insufficient data. In Table 4.2, we observe that our method still outperforms its baseline overall. Although it dominates with random buffers and has comparable performance with expert buffers, it does not learn well with medium buffers. The root cause is the similarity of the quality of actions between medium buffers and RBC policy, which causes the critic to misjudge which action to pick between them. However, RUBICON still outperforms the baseline in other two types of buffers since the value estimation differences between $(\pi_b(s), s)$ and $(\pi_{rbc}(s), s)$ are more distinguishable in these scenarios.

**Policy analysis experiment** Since Q-value prediction is usually overestimated, we use immediate rewards as references to examine the quality of the inferences of Q-networks. We pre-train a reward model $R_\psi(s, a)$ using the data in the buffer to predict reward $\hat{r}$ given state $s$ and selected action $a$ with 200K iterations with the buffer as our training data. At each iteration of the policy update, we record the policy $\pi(s)$ and the predicted rewards in each batch, i.e., $\hat{r} = R_\psi(s, \pi(s))$. We plot the distributions of reward in action spaces in Fig. 4.7. It demonstrates that RUBICON selects the actions in a wider range cf. TD3+BC, nonetheless, with a reward distribution of higher values. The distribution shown is with 10% of data randomly selected from the entire training for better visualization.

**Table 4.2**: Data reduction experiment

| Environment | Buffer | RUBICON | TD3+BC |
|---|---|---|---|
| hot-deterministic | Expert | **89.65±9.12** | 74.96±12.38 |
| hot-deterministic | Medium | -6.92±78.43 | **79.37±25.5** |
| hot-deterministic | Random | **90.41±5.47** | -24.18±59.62 |
| mixed-deterministic | Expert | 43.09±60.36 | **93.77±8.71** |
| mixed-deterministic | Medium | 48.24±54.32 | **98.67±2.28** |
| mixed-deterministic | Random | **82.89±10.19** | 72.58±6.91 |
| cool-deterministic | Expert | **89.52±11.28** | 86.45±8.62 |
| cool-deterministic | Medium | -11.56±62.82 | **47.61±17.66** |
| cool-deterministic | Random | 35.92±43.24 | **42.59±72.43** |
| hot-stochastic | Expert | 80.35±25.8 | **89.02±5.51** |
| hot-stochastic | Medium | 5.3±43.33 | **10.79±56.47** |
| hot-stochastic | Random | **62.26±18.59** | 26.43±52.33 |
| mixed-stochastic | Expert | 78.08±25.78 | **85.25±12.96** |
| mixed-stochastic | Medium | 66.02±17.68 | **72.17±16.14** |
| mixed-stochastic | Random | **55.26±29.19** | -86.21±25.92 |
| cool-stochastic | Expert | **98.18±2.56** | 39.04±86.35 |
| cool-stochastic | Medium | **73.16±19.56** | 38.68±39.93 |
| cool-stochastic | Random | **69.76±5.46** | -49.5±9.75 |
| Sum | . | **1049.61±523.18** | 797.49±519.47 |

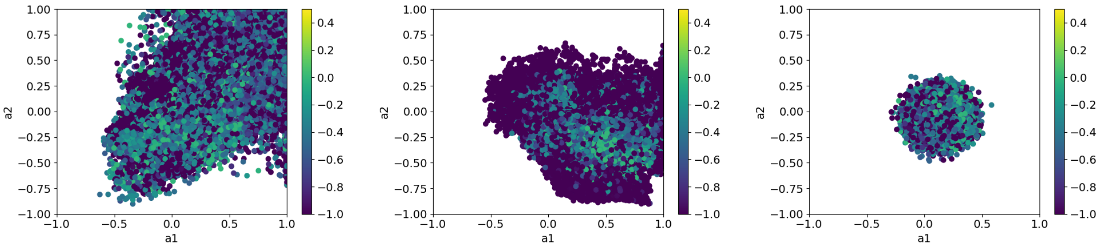


**Figure 4.7**: Reward distribution in action spaces of *hot-continuous* environment learns from medium buffer, from left to right: RUBICON (1.842/1.978/-0.577), TD3+BC (1.534/1.332/-0.668), and buffer (0.908/0.915/-0.799); tuples are the values of (action1 range/action2 range/reward mean).

## 4.5.2 Online approach

### Main Experiment

In the online approach, it is assumed that an oracle exists for accurate simulations. In real-world applications, researchers train online models in simulation environments before deployment in real buildings. Experimental results comparing TD3 and our method can be found in Table 4.3. In five out of six tasks, our method outperforms TD3 in averaged scores and with a substantially smaller standard deviation across runs. The learning curves are illustrated in Fig. 4.8. In terms of the optimization objectives, RUBICON yields a 13.16% of energy reduction and 17.86% reduction in comfort penalty.

These results empirically show that our method strengthens the learning process not only in the offline approach but also in the online approach.

**Table 4.3**: Online RUBICON and TD3 comparison

| Environment | RUBICON | TD3 |
|---|---|---|
| hot-deterministic | **79.08±12.24** | 51.83±49.93 |
| mixed-deterministic | **72.34±0.00** | 23.46±41.01 |
| cool-deterministic | **66.52±0.00** | **66.3±41.66** |
| hot-stochastic | **83.64±8.25** | 71.8±21.51 |
| mixed-stochastic | **72.24±0.49** | 8.5±66.61 |
| cool-stochastic | 68.14±0.65 | **73.38±16.61** |
| Sum | **441.99±21.64** | 295.29±237.36 |

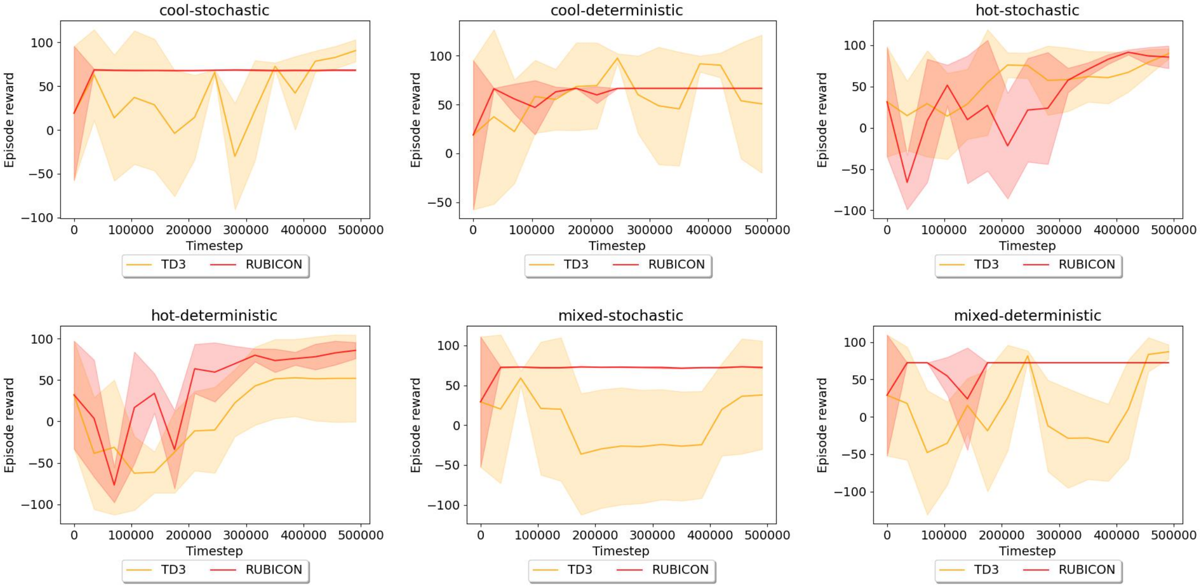


**Figure 4.8**: Learning curves comparing online RUBICON and TD3

**Hyperparameter experiment**

Deep-RL is sensitive to hyperparameter tuning [3], thus, we keep the original neural network architectures and hyperparameter settings for a fair comparison. Since all the authors of these methods have optimized the hyperparameters across various tasks and randomly initialized conditions. However, for the online settings, we introduce the behavioral cloning term in the actor loss. We conduct hyperparameter optimization experiments for the optimized settings. When the behavioral policy's mean Q-value of the batch is higher than RBC policy, the actor loss follows the original TD3 algorithm. On the opposite, with RBC policy having a higher mean Q-value, the weighting $\lambda$ and its hyperparameter $\alpha$ (see Eq. 4.2 and Alg. 3) should be optimized since we cannot assume the model's behavior is similar to the offline setting. It is mentioned in the TD3+BC paper that the value of $\alpha$ decides if the model learns similarly to RL ($\alpha$=4) or imitation learning ($\alpha$=1) and the default value set in TD3+BC is $\alpha$ =2.5. We experiment on the values $\{1, 2.5, 4\}$ to observe how it affects the performance of our models in all tasks. The result (See Table B.1) shows that when $\alpha$=1 the model gives the highest scores and the least variance. This indicates that the agent should imitate RBC policy even more than the offline setting ($\alpha = 2.5$) in order to achieve a more optimal policy.

## 4.6 Conclusion and Future Works

In this paper, we explored how rule-based control policies can be incorporated into reinforcement learning as regularization to improve both of their performance. Our method can be implemented on the baseline methods with minimal changes and is straightforward and intuitive. We applied our method in building HVAC control simulation environments in both online and offline settings, demonstrating its practical usage regardless of the existence of a valid environment simulator. We empirically demonstrate that our method outperforms state-of-the-art offline/batch reinforcement learning methods and improves from its online baseline by a substantial amount in building HVAC control tasks where rule-based control is robust and a standard in real-world settings. We expect our study, open-sourced code bases and dataset[2] would encourage both domains and RL experts to explore more opportunities for the combination of existing policies and RL and extend this concept to more real-world applications.

For future works, we plan to enhance the interpretability of the decision-making process in our experiments, we aim to develop transparent and interpretable algorithms for RL agents via Explainable RL (XRL) [107, 1, 46]. Also, using the ensemble Q-networks for a more accurate Q-value estimation.

Chapter 4, in part, is a reprint of the material that appears in the proceeding of the 14th ACM International Conference on Future Energy Systems (e-Energy 2023). By authors Hsin-Yu Liu, Bharathan Balaji, Rajesh Gupta, and Dezhi Hong with the title - “Rule-based Policy Regularization for Reinforcement Learning-based Building Control” The dissertation author is the primary investigator and author of this paper.

[2] For the dataset please refer to our B2RL (`https://github.com/HYDesmondLiu/B2RL`) repository. And for the codes please refer to our RUBICON (`https://github.com/HYDesmondLiu/RUBICON`) repository.

# Chapter 5

# Adaptive Policy Regularization for Offline-to-Online Reinforcement Learning in HVAC Control

## 5.1 Introduction

In Chapter 2, we considered offline RL approaches for the scenarios where we have no simulators of the environments, but the data of sensing and actuating. In Chapter 3, we open-source the first dataset for building the BRL learning benchmark. In Chapter 4, we incorporate the existing policies with RL. Now we build upon these to further improve the pre-trained offline models via online interactions.

Real-world building HVAC systems are mainly controlled via Rule-Based Control (RBC) policies that consist of a set of if-else rules crafted by domain experts. However, it is

**Table 5.1**: Comparison between RL approaches

| RL approaches | Strength | Weakness | Prerequisite |
|---|---|---|---|
| Online | Ability to explore and earn directly from real-time interactions | Learning from random policies | Access to a live environment for interaction |
| Offline | Learn from a fixed dataset, reducing risk | Large and diverse dataset of good-quality experiences | Limited to the diversity of the dataset |
| Offline-to-Online | Combining the safety of offline data with the adaptability of online updates | Complexity in integrating offline and online learning phases | Access to both historical data/pre-trained offline models and a live environment |

unlikely for these rules to generalize and scale to different operating environments and buildings. Reinforcement Learning, on the other hand, can adapt to the changes in the environment. To do so, building control problems could be formulated as Markov Decision Processes (MDP) and to be optimized with RL methods [120]. The RL agent observes the states via sensors in the buildings, (i.e., thermostats, hygrometers), and the actions are implemented as the control setpoints in a building management system via actuators. In this formulation, the objective of the agent is to increase energy efficiency while ensuring occupants' thermal comfort [145].

Most RL methods in prior works are trained and evaluated in the online paradigm. With the online approach, an accurate simulator is required to give feedback to the RL agent. However, setting and calibrating these simulators requires expertise and is time-consuming. Also, in modern commercial buildings, the building management systems already store a significant amount of sensing and control data. These datasets capture not only the thermal dynamics of the systems but also the control policies that embody the best-known knowledge of the control defined by domain experts. Prior works have shown that offline data-driven methods can learn a more robust policy than the behavioral RBC policies that generated the data [68]. Nonetheless, without further online interactions, the improvement is limited by the diversity of the state-action distribution in the historical data.

Recently, researchers have explored the ability of offline-to-online (O2O) RL in other domains [78, 148, 142, 138, 62]. These models are designed to adapt to the distribution drift between the static buffers and the environments evaluated. However, most of these methods deteriorate the performance of the pre-trained offline models. In this chapter, we aim to answer the following research questions:

**Q.1** ***How can we adapt to distribution drifts under offline-to-online paradigm, without harming the capability of the pre-trained models?***

**Q.2** ***How can we further improve the pre-trained models as training evolves?***

**Q.3** ***How much historical data is required for online, offline, and offline-to-online methods to learn a robust and stable policy?***

These questions are crucial not only for O2O RL methods but also for real-world building control problems, considering both the capability of the O2O models and data efficiency. The objective of O2O RL is to adapt to the building environment with limited data, and simultaneously maintain the pre-trained performance.

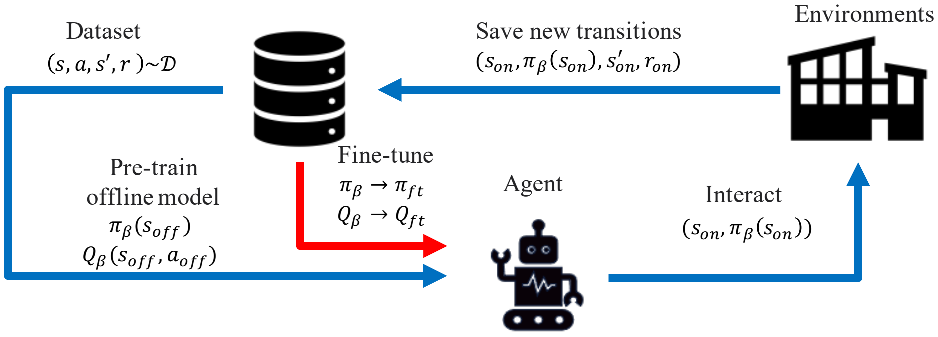

Dataset
$(s, a, s', r) \sim \mathcal{D}$
Save new transitions
$(s_{on}, \pi_\beta(s_{on}), s'_{on}, r_{on})$
Environments
Pre-train
offline model
$\pi_\beta(s_{off})$
$Q_\beta(s_{off}, a_{off})$
Fine-tune
$\pi_\beta \rightarrow \pi_{ft}$
$Q_\beta \rightarrow Q_{ft}$
Agent
Interact
$(s_{on}, \pi_\beta(s_{on}))$

**Figure 5.1**: Flow chart of offline-to-online RL: (1.) The offline model learns from the existing dataset. (2.) After pre-training, the agent interacts with the environment online. (3.) The generated transitions are saved in the replay buffer(s) for further learning. (4.) The offline-to-online fine-tuning improves the agent's performance continuously.

## 5.2 Related Work

We build up prior work on offline-RL, O2O-RL, and building RL controls. We summarize the comparison between different RL approaches in Table 5.1.

### 5.2.1 Building RL control

Prior research has demonstrated that building RL control policy could outperform RBC in both online and offline settings. Zhang et. al. [145] developed a framework for whole building HVAC (heating, ventilation, air-conditioning) control in online settings to achieve a 16.7% heating demand reduction cf. RBC control. OCTOPUS holistically controls subsystems in modern buildings to get a 14.26% energy saving cf. RBC policy [23]. Yang et. al. [135] implemented an RL control for LowEx building systems with a 11.47%

improvement on cumulative net power output than RBC.

With offline RL, Zhang et. al. [141] applied a state-of-the-art method and demonstrated a $12 \sim 35\%$ of reduction in ramping. In Chapter 2, we incorporated a Kullback-Leibler (KL) divergence constraint during the training of an offline RL agent to penalize policies that are far away from the previous updates for stability, and achieve a 16.7% of energy reduction cf. the default RBC control. To our knowledge, there is no O2O-RL study in the building control domain. We are the first to study O2O-RL via building HVAC control.

### 5.2.2 Offline RL

In real-world settings, we often have access to data generated with an existing 'behavioral' policy (RBC policy in our work) when there are no established simulators of the environment. These logged interactions are saved as experience replay or replay buffers. Offline RL learns exclusively from existing static datasets without interacting with an environment. Due to the lack of accurate value estimation of out-of-distribution (OOD) state-action pairs, these methods learn a more conservative policy or a pessimistic lower bound of the value estimation.

BCQ [37] mitigates the extrapolation errors induced by OOD actions via a variational autoencoder. BEAR [57] uses ensemble Q-functions to reduce the accumulation of bootstrapping errors. BRAC [133] regularizes the learned policy towards the behavioral policy with a KL divergence constraint between the distributions over actions. CQL [59] learns a lower bound of the true Q-function with SAC [39]-style entropy regularization. TD3+BC [33], derived from TD3 [35], uses a behavioral cloning regularization for policy learning. UWAC [134] down-weights the OOD state-action pairs' contribution to the training.

### 5.2.3 Offline-to-Online RL

Offline-to-online (O2O) RL follows the assumption of offline RL where there is no access to the simulator of the system. However, we could further improve the model with online interactions since the pure offline method cannot yield accurate value estimation of the OOD state-action values. Hence, the goal is to enhance the capability of the model with online training without learning from scratch as in the traditional online setting.

#### RL with Offline Data

Previous studies focus on online RL boosted with offline data. One branch in this research area is RL with Expert Demonstrations (RLED) with the assumption that a pre-trained offline model may not be necessary. APID (Approximate Policy Iteration with Demonstrations.) [49] leverages few and/or sub-optimal demonstration data used as suggestions to guide the optimization performed by approximate policy iteration. DQfD (Deep Q-learning from Demonstration) [44] leverages demonstration data to accelerate the online learning process. Piot et al. [84] proposes a method to minimize the optimal Bellman residual guided by constraints defined by the expert demonstrations. Recently, RLPD (Reinforcement Learning with Prior Data) [7] extends standard off-policy RL and achieves state-of-the-art online performance on a number of tasks using offline data not limited to expert prior knowledge [8]. This branch of research differs from our study in that we focus on fine-tuning the pre-trained offline models and not training the models from scratch with offline data to accelerate the learning.

### Online fine-tuning with offline pre-training

Another branch, which is similar to our proposed method, assumes we fine-tune offline models in online settings to adapt to distribution drift and optimize the exploration-exploitation trad-offs. AWAC [78] trains an advantage-weighted actor-critic with an implicit policy constraint to avoid with a balanced replay between offline and online buffers, a pessimistic Q-ensemble [62], and a density ratio estimator to improve sample efficiency and prevent over-optimism. PEX [142] freezes the pre-trained offline model, expands the policy set with the fine-tuning model, and constructs a categorial distribution for selecting the final action. APL [149] obtains near-on-policy data and chooses an optimistic update strategy. On the other hand, it uses a pessimistic update strategy for sampled offline data. REDQ [148] uses randomized ensemble Q-functions to increase sample efficiency and adaptive hyperparameter tuning to adjust the degree of behavioral policy regularization with a normalized target episode reward. ACA [138] introduces a reconstruction of Q-functions for online fine-tuning as an alignment step so it is tamed to be consistent. TD3-C [72] considers conservative policy optimization as the approach for stabilizing finetuning when the offline dataset lacks diversity.

Unfortunately, the aforementioned offline-to-online methods need at least one of the following requirements that makes them resource-consuming [138, 148, 62, 78, 72] and most of them fail to maintain the pre-trained performance:

**R.1** Introducing additional models other than existing ones and/or maintaining multiple buffers.

**R.2** Require information on absolute scores of expert and random agents that may not be accessible.

**R.3** Many suffer policy collapse at the very beginning of the transition from offline mode to online mode.

To summarize, the key contributions of our work are:

- Our method requires no extra models, only a single replay buffer and works without the identifying the offline transitions and the online transitions (to tackle **R.1**).
- Develop an automatic fine-tuning offline-to-online RL algorithm that could maintain the pre-trained model's ability and continue to learn to reach an optimal policy (to tackle **R.2** and **R.3**).
- The add-on methods - Combined Experience Replay (CER [143]) and Bootstrapped Ensemble [80] help adapt the distribution drift with extreme low cost of $\mathcal{O}(1)$ time complexity.

The details of our method will be elaborated in Sec.5.4.

## 5.3 Problem Formulation

### 5.3.1 Reinforcement Learning

Reinforcement Learning problems are formulated as a Markov Decision Process (MDP), a sequential decision-making problem that aims to maximize the discounted accumulative rewards. The MDP consists of a tuple: $\mathcal{M} = (\mathcal{S}, \mathcal{A}, p, r, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathbb{P}$ is the transition dynamics. The next state $s_{t+1} \sim p(\cdot|s_t, a_t)$, is decided by the current state and the action selected by a policy $\pi(a|s), \pi : \mathcal{S} \rightarrow \mathcal{A}$ either in a stochastic or a deterministic fashion. The reward function $R : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}, r \in \mathbb{R}$ is mapped as a scalar, and the discount factor $\gamma \in [0, 1)$. The agent's goal is to optimize the policy to maximize the discounted accumulated return $\mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t r_t]$ [109].

### 5.3.2 Offline RL training

In offline training, the replay buffer $\mathcal{D}$ is generated by an unknown behavioral policy (or a combination of multiple policies): $\pi_\beta(s)$. Then the offline model aims to learn the optimal policy without interacting with the environment within the confined state-action visitations. Thus, when the trained offline RL policy is deployed in the real environment, any OOD actions may lead to inaccurate value estimation due to extrapolation errors.

Our offline training follows TD3+BC, an offline version of TD3 [35] with minimal modification from its online alternative. The learned policy is regularized with a behavior cloning term to penalize policies that deviate from the behavioral policy:

$$\pi = \underset{\pi}{\operatorname{argmax}} \, \mathbb{E}_{(s,a)\sim\mathcal{D}}[\lambda \bar{Q}(s, \pi(s)) - (\pi(s) - \pi_\beta(s))^2] \tag{5.1}$$

$$\lambda = \frac{\alpha}{\frac{1}{N}\sum_{(s_i,a_i)} |Q(s_i, a_i)|}$$

Where N is the size of the minibatch, $\bar{Q}$ is the average Q-value in the sampled batch given $s$ and $\pi(s)$, $\pi_\beta(s)$ denotes the behavioral policy given $s$, and $\alpha$ is a hyperparameter to balance between the online exploration and the exploitation of the behavioral policy.

## 5.4 Methodology

### 5.4.1 Offline-to-Online RL Training via Weighted Increased Simple Moving Average Q-value

#### Observation and Insight

Our method WISMAQ (Weighted Increased Simple Moving Average of Q-value) is inspired by the preliminary experiments we performed with the B2RL [69] RBC buffers with pre-trained TD3+BC (pure offline) then converted to TD3 (pure online) during offline-to-online fine-tuning.

Specifically, we add up all the averaged Q-values of the sampled batches at each timestep $i$ for $\bar{Q}_{\mathcal{B}_i}$, the mean of the average Q-value sampled from the batches. Then, when an episode ends at time $t_e$ we store the episodic average $\bar{Q}_e = \frac{1}{t_e}\sum_{i=1}^{t_e} \bar{Q}_{\mathcal{B}_i}$. Finally,

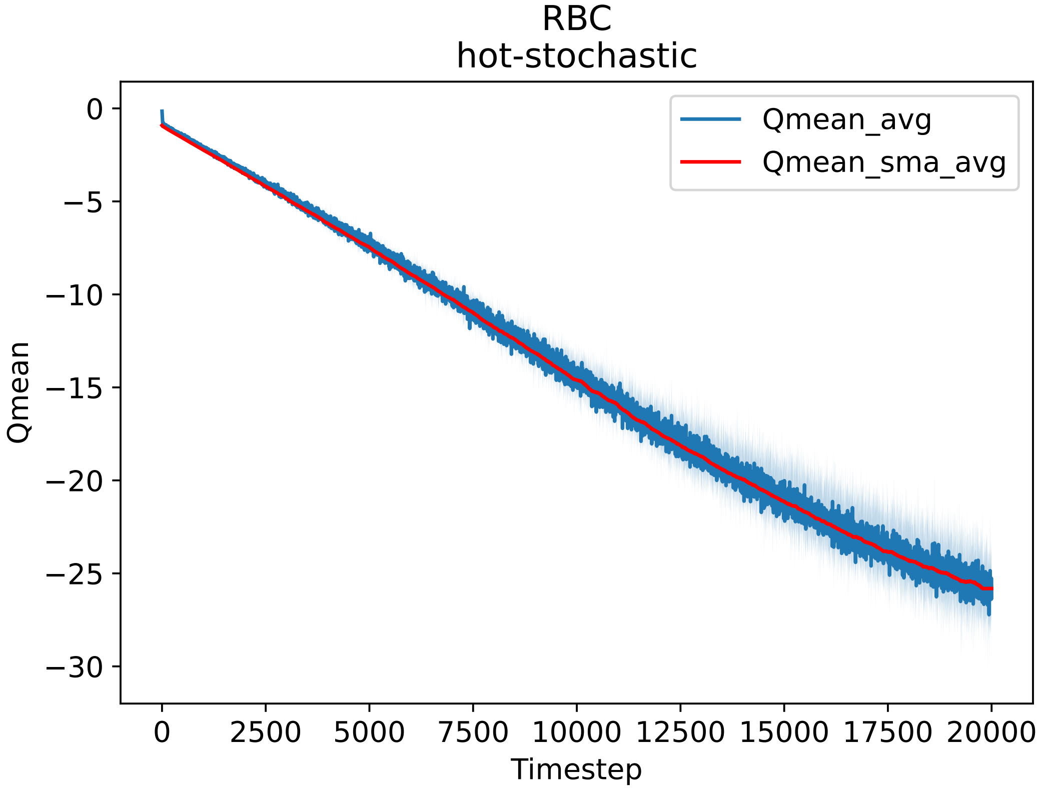


**Figure 5.2**: The averaged Q-value of the batches sampled from the RBC buffer during the training, the agent failed to improve its policy, thus the mean Q-value converges.

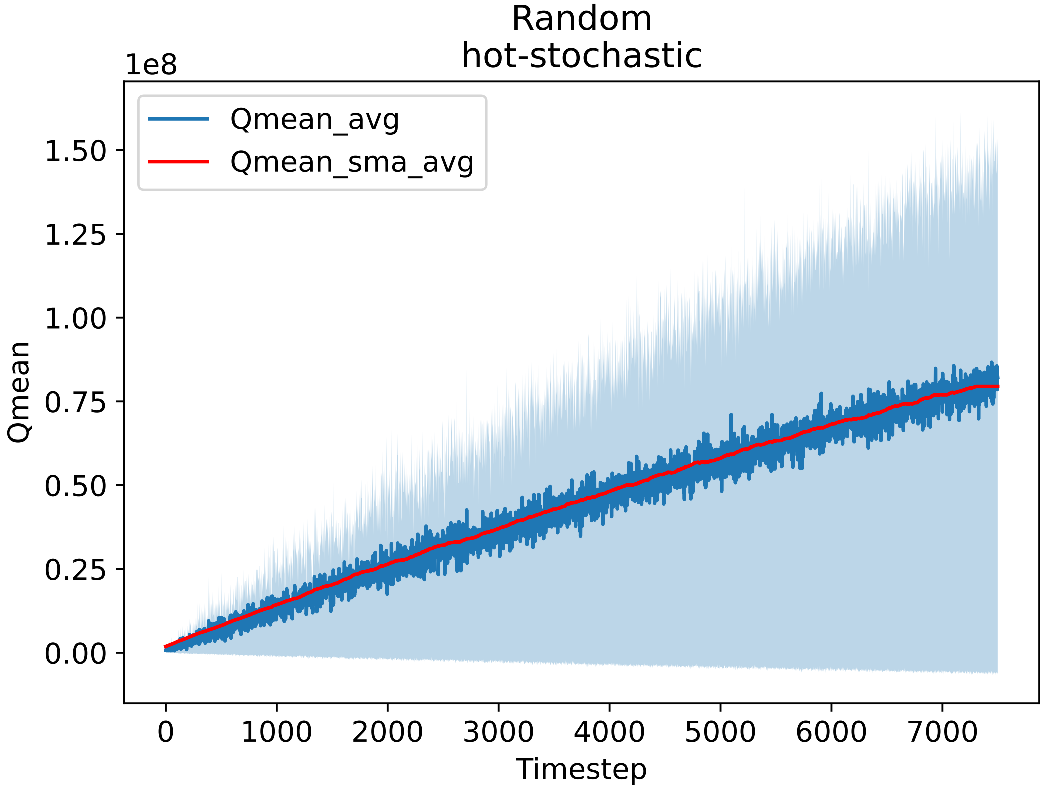


**Figure 5.3**: The averaged Q-value of the batches sampled from the random buffer during the training, the agent learns better policies, the mean Q-value is increased over time.

we plot the average episodic mean Q-values over time in Fig. 5.2 (where the solid blue curves are the average values and the shaded regions are the min/max range), as the red curve demonstrates the Q-SMA (Simple Moving Average of mean Q-value sampled from the batches).

As we can see in Fig. 5.2, the averaged Q-value of the sampled batch is noisy due to the random sampling from the buffers. However, the Q-SMA is more stable cf. the raw average Q-value. In this example, the agent fails to learn a better policy, thus, the Q-value converges as training continues. On the opposite, in Fig. 5.3, the agent continues to learn better policies as training goes on. Thus, the average Q-value increases as the improved policy has generated transitions with higher Q-values to be stored in the replay buffer.

The agent learns a better policy will generate the transitions that are with higher values. And it could be validated by the estimations from the critics, i.e., higher Q-values. According to Eq. 5.1 it indicates the policy leads to a higher average Q-value with the sampled states and actions selected by the policy. We observe that the average Q-value increases as training proceeds for buffers with lower behavioral policy performance. For the expert task, the model is unable to learn a better policy to yield a higher estimated Q-value during the training, i.e., the value estimation converges [104].

Our method WISMAQ is simple and straightforward – it aims to learn a policy that yields an increasing simple moving average (SMA) Q-value in the sampled batch compared to a previous reference SMA. We use SMA instead of the vanilla average since the average Q-value in the batch is noisy due to random sampling and inherited uncertainty within the models. The regular average does not reflect the actual trend of mean Q-value. The timestep difference between the current SMA and the reference SMA is a hyperparameter to be optimized; it depends on how rapidly the value estimation varies. If we simply use greedy Q-value increments, it would lead to abrupt performance collapse when Q-value estimation is uncertain and encounters unseen environment state-action distributions.

### Averaged Q-value in Replay Buffers

Based on the observation in the previous subsection, we observe that when the agent could not learn a better policy the average Q-value in the replay buffer is lowered as training continues. However, if the agent can learn a better policy, the average Q-value in the replay buffer would increase as training continues. It could be proved by the following:

With policy improvement, the value of any state $s$ correspondingly improves based on the standard policy improvement theorem.

**Lemma 1**: Policy Improvement Theorem

***Proof*** *[104]*:

Given a policy $\pi$

$$v_\pi(s) = \mathbb{E}[R_{t+1} + \gamma R_{t+2} + ...|S_t = s]$$

$\pi' = greedy(v_\pi)$. Consider a deterministic policy $a = \pi(s)$, improve it by acting greedily with respect to $v_\pi$

$$\pi'(s) = \operatorname*{argmax}_{a \in \mathcal{A}} q_\pi(s, a)$$

it improves the value from any state $s$ over one step:

$$q_\pi(s, \pi'(s)) = \max_{a \in \mathcal{A}} q_\pi(s, a) \geq q_\pi(s, \pi(s)) = v_\pi(s)$$

Hence, the value of any state $s$ is improved, i.e., $v_{\pi'}(s) \geq v_\pi(s)$

$$v_\pi(s) \leq q_\pi(s, \pi'(s)) = \mathbb{E}_{\pi'}[R_{t+1} + \gamma v_\pi(S_{t+1}) | S_t = s]$$

$$\leq \mathbb{E}_{\pi'}[R_{t+1} + \gamma q_\pi(S_{t+1}, \pi'(S_{t+1})) | S_t = s]$$

$$\leq \mathbb{E}_{\pi'}[R_{t+1} + \gamma R_{t+2} + \gamma^2 q_\pi(S_{t+2}, \pi'(S_{t+2})) | S_t = s]$$

$$\leq \mathbb{E}_{\pi'}[R_{t+1} + \gamma R_{t+2} + ... | S_t = s] = v_{\pi'}(s)$$

We now prove that an improved policy yields a higher average Q-value in the replay buffer.

**Lemma 2**: An improved policy will yield a higher average Q-value in the replay buffer as we store the newly generated transitions into it.

***Proof***: The Q-value update rule for the tabular Q-learning algorithm [127] is:

$$Q_{t+1}(s, a) \leftarrow Q_t(s, a) + \alpha[r + \gamma \max_{a'} Q_t(s', a') - Q_t(s, a)]$$

Assuming the old policy is $\pi_t$ and the newly improved policy is $\pi_{t+1}$, then the Q-value of the new policy can be expressed as:

$$Q^\pi_{t+1}(s, a) = \mathbb{E}[r + \gamma \max_{a'} Q^\pi_{t+1}(s', a') | s, a, \pi_{t+1}]$$

$$= \sum_{s'} P(s' | s, a, \pi_{t+1})[r + \gamma \max_{a'} Q^\pi_{t+1}(s', a')]$$

where $P(s'|s, a, \pi_{t+1})$ is the transition probability. By the definition of the Q-value and the assumption that a new policy is better than the old one, we have: $Q^\pi_{t+1}(s, a) \geq Q^\pi_t(s, a)$.

$\forall s \in \mathcal{S}$ and $\forall a \in \mathcal{A}$. Taking the average of both sides over all state-action pairs, we have:

$$\frac{1}{|\mathcal{S}||\mathcal{A}|}\sum_{s\in\mathcal{S}}\sum_{a\in\mathcal{A}}\mathbb{E}[r+\gamma\max_{a'}Q^{\pi}_{t+1}(s',a')|s,a,\pi_{t+1}] \geq$$

$$\frac{1}{|\mathcal{S}||\mathcal{A}|}\sum_{s\in\mathcal{S}}\sum_{a\in\mathcal{A}}Q^{\pi}_{t}(s,a)$$

Since the left-hand side is the average Q-value of the new policy over all the state-action pairs and the right-hand side is the average Q-value of the old policy over all state-action pairs, this inequality shows that the new policy yields transitions with a higher averaged Q-value than the averaged Q-value of the transitions generated by the old policy. End of proof.

**WISMAQ Implementation**

As we can observe in Fig. 5.2, the trace of the episodic average Q-value is noisy while the Q-SMA is more stable. Thus, we introduce the simple moving average of the average Q-values to yield a more statistically meaningful metric for the model. To apply our method on TD3+BC, we modify the policy update from Eq. 5.1 with the added loss term $\mathcal{L}_{WISMAQ}$:

$$\mathcal{L}_{WISMAQ} = ReLU\left(\frac{\bar{Q}^{t}_{SMA} - \psi * \bar{Q}^{t-d}_{SMA}}{\bar{Q}^{t}_{SMA} + \bar{Q}^{t-d}_{SMA}}\right) \tag{5.2}$$

where $t$ is the current timestep, $\psi$ is a hyperparameter to weight the reference Q-SMA, and $d$ is the difference between the current timestep and the reference timestep. ReLU is the rectified linear unit activation function that automatically tunes this term based on the difference of the Q-SMA between the reference timestep and the current timestep. i.e., when $\bar{Q}^{t}_{SMA}$ is larger than $\psi * \bar{Q}^{t-d}_{SMA}$, this term is activated. On the opposite, when

$\bar{Q}^t_{SMA}$ is smaller than $\psi * \bar{Q}^{t-d}_{SMA}$, this term is deactivated. The model learns the same as the offline models (see Eq.5.1).

And the SMA (Simple Moving Average) for the timestep $t$:

$$\bar{Q}^t_{SMA} = \frac{\bar{Q}^t + \bar{Q}^{t+1} + ... + \bar{Q}^{t+w}}{w} \tag{5.3}$$

where $w$ is the window size we use for calculating the SMA. Thus, the policy update follows:

$$\pi = \underset{\pi}{\operatorname{argmax}} \, \mathbb{E}_{(s,a)\sim\mathcal{D}}[\lambda\bar{Q}(s,\pi(s)) - (\pi(s) - \pi_\beta(s))^2 + \xi\mathcal{L}_{\mathbf{WISMAQ}}] \tag{5.4}$$

And $\xi$ is a hyper-parameter for the coefficient of $\mathcal{L}_{WISMAQ}$. With the ReLU activation function, the added loss term is bounded, i.e. $\mathcal{L}_{WISMAQ} \in [0, 1]$. And thus makes it auto-tuned without any heuristics.

**To summarize**, WISMAQ optimizes the policy by identifying if the current SMA of the averaged Q-value is higher than a previous reference SMA. If so, the WISMAQ loss term is activated, it encourages the policy to explore. Otherwise, when the WISMAQ loss term is lower than the reference value, we keep the actor loss as is since it means the Q-value is converging, i.e., leave it learning as the original offline model, which is conservatively trained with online transitions.

### 5.4.2 Adapting to Distribution Drift

Another critical challenge in O2O transition is the distribution drift as building and environmental conditions change. Previously, several methods were proposed to accelerate RL learning by utilizing replay buffers [143, 98, 28]. We found some of them are suitable to deal with the distribution drift in the O2O setting. The first one we adopt is the combined experience replay (CER) [143], it adds the latest transition to the sampled batch in every training step and speed up the learning. Intuitively, it forces the model to learn from the latest state-action distribution of the environment combined with the previous ones.

The other technique we applied is to remove the oldest transition in the buffer faster by setting a smaller number of transitions stored in the replay buffer since it is implemented in queues [28]. When a policy is learning and improving, the transitions generated by the old policies might harm the convergence of the model due to its inferior performance cf. the current policy. Especially in off-policy settings, we learn from the behavioral policy via replay buffer. Observations from our experiments (in the next section) indicate that the performance of the models consistently improves with the reduced age of the oldest policy [28]. The CER technique could be implemented with minimal changes with only $\mathcal{O}(1)$ time complexity. Further setting a smaller buffer size requires no modification of the algorithm itself, which is efficient and reasonable in O2O training. We detail the steps of our method in Algorithm 5.

### 5.4.3 Bootsrapped Ensemble Learning

Due to the need for exploring uncharted state-action spaces, almost all previous O2O studies take advantage of certain kinds of ensemble learning. We use $K$ bootstrapping ensemble double-Q networks via different combinations of the randomly sampled batches in each iteration of critic training, then we randomly select a value network at each policy training iteration. This is inspired by the bootstrapped DQN for deep exploration [80].

**Algorithm 5:** WISMAQ offline-to-online fine-tuning

```
Load pre-trained offline model as $K$ ensemble double-Q networks $\{Q_{i,\theta_1}, Q_{i,\theta_2}\}_{i=1}^{K}$,
  the actor-network $\pi_\phi$, with random parameters $\{\theta_{i,1}, \theta_{i,2}\}_{i=1}^{K}$, $\phi$, target networks
  $\{\theta'_{i,1} \leftarrow \theta_{i,1}, \theta'_{i,2} \leftarrow \theta_{i,2}\}_{i=1}^{K}$, $\phi' \leftarrow \phi$, policy update frequency $f$, horizon $T$, replay
  buffer $\mathcal{B}$
for $t = 0$ to $T$ do
    Select actions with exploration noise
    $a \sim \pi_\phi(s) + \epsilon$, $\epsilon \sim \mathcal{N}(0, \sigma)$
    Observe reward $r$ and next state $s'$
    Store transition $t = (s, a, r, s')$
    Delete the oldest one in $\mathcal{B}$ for $i = 1$ to $K$ do
        Sample $N$ transitions $(s, a, r, s')$ from $\mathcal{B}$ in which $t \subseteq \mathcal{B}$ (CER)
        $\tilde{a} \leftarrow \pi_{\phi'}(s') + \epsilon$, $\epsilon \sim \text{clip}(\mathcal{N}(0, \tilde{\sigma}), -c, c)$
        $y \leftarrow r + \gamma \min_{j=1,2} Q_{\theta'_{i,j}}(s', \tilde{a})$
        Update critics
        $\theta_{i,j} \leftarrow \text{argmin}_{\theta_{i,j}} N^{-1} \sum (y - Q_{\theta_{i,j}}(s, a))^2$
    if $t$ mod $f$ then
        Update $\phi$ by policy gradient:
        Policy update follows Eq. 5.2, 5.3, and 5.4(Bootstrapped Ensemble-Q and
          WISMAQ)
        Calculate $\nabla_\phi J(\phi)$
        Update target networks:
        for $i = 1$ to $K$ do
            $\theta'_{i,j} \leftarrow \tau\theta_{i,j} + (1 - \tau)\theta'_{i,j}$
        $\phi' \leftarrow \tau\phi + (1 - \tau)\phi'$
```

The authors claim that randomized value functions offer a promising approach to efficient exploration with generalization. Our experiment results agree with the statement with fewer variances across different runs and could learn faster than other methods.

## 5.5 Benchmark Experiments

### 5.5.1 Experiment Setups

We conducted our experiments with the building RL simulation environments [47]. The objective of the agent is to maintain a comfortable thermal environment with minimal energy use. The state consists of indoor/outdoor temperatures, time/day, occupant count, thermal comfort, and related sensor data. The action adjusts the temperature setpoint of the thermostat. The reward is a linear combination of occupants' thermal comfort and energy consumption. The environment is a single-floor building divided into 5 zones, with one interior and four exterior rooms.

The details about the RL settings in our problem are described below:

- **State**: Site outdoor air dry bulb temperature, site outdoor air relative humidity, site wind speed, site wind direction, site diffuse solar radiation rate per area, site direct solar radiation rate per area, zone thermostat heating setpoint temperature, zone thermostat cooling setpoint temperature, zone air temperature, zone thermal comfort mean radiant temperature, zone air relative humidity, zone thermal comfort clothing value, zone thermal comfort Fanger model PPD (predicted percentage of dissatisfied), zone people occupant count, people air temperature, facility total HVAC electricity demand rate, current day, current month, and current hour.

- **Action**: Heating setpoint and cooling setpoint in continuous settings for the interior zones.

- **Reward**: We follow the default linear reward setting, which considers the energy consumption and the absolute difference to temperature comfort.

  The reward function is described below:

  $$r_t = -\omega \lambda_P P_t - (1-\omega)\lambda_T(|T_t - T_{up}| + |T_t - T_{low}|) \tag{5.5}$$

  where $P_t$ represents power consumption; $T_t$ is the current indoor temperature; $T_{up}$ and $T_{low}$ are the imposed comfort range limits (penalty is 0 if $T_t$ is within the range); $\omega$ is the weight assigned to power consumption. Finally, $\lambda_P$ and $\lambda_T$ are scaling constants for energy consumption and comfort, respectively.

- **Environment**: A single floor building with an area of $463.6m^2$ divided into 5 zones, 1 interior, and 4 exteriors. The HVAC system is a packaged VAV (variable air volume) (DX (direct expansion) cooling coil and gas heating coils) with fully auto-sized input. And the simulation period of one episode is a full year. The weather types are classified according to the U.S. Department of Energy (DOE) standard [79]. The weather type details and their representative geometric locations are listed below based on TMY3 datasets [60]:

- **Cool marine**: Washington, USA. The mean annual temperature and mean annual relative humidity are 9.3°C and 81.1% respectively.
- **Hot dry**: Arizona, USA with mean annual temperature of 21.7°C and a mean annual relative humidity of 34.9%
- **Mixed humid**: New York, USA with a mean annual temperature of 12.6°C and a mean annual relative humidity of 68.5%

### 5.5.2 Experiments

We create a set of RBC buffers for the three different weather conditions. Considering two practical scenarios where we have a well-trained RL agent and an RBC policy written by human experts. This is the general case for most large commercial building sectors. The generation flow of the buffers is described in detail below:

(1.) Generate buffers for $250K$ time steps via Rule-Based Control policy:

- **RBC**: We follow the SinerGym [47]'s example of RBC policy and vectorize it for better computation efficiency (see Alg. 6).

(2.) Every method is trained offline first for $50,000$ timesteps with the RBC buffers.

(3.) Train the state-of-the-art methods along with WISMAQ as the flow in Fig. 5.1 to compare the learning curves. We trained the models with $35,000$ timesteps which is approximately one year in real-time. And the evaluation frequency is each $2,500$ timesteps. Each algorithm is run with three random initialization setups (3 random seeds).

We list below the methods compared:

**Algorithm 6:** Rule-based control policy

```
Input  : Current datetime $current\_dt$, indoor air temperature $IAT$, zone
         thermostat heating setpoint temperature $a_h$, and zone thermostat
         cooling setpoint temperature $a_c$, obtained from the states with size $N$
Output: Actions selected by RBC
for $i$ in $N$ do
    $season\_comfort\_zone_i = get\_season\_comfort(current\_dt_i)$
    if $IAT_i >= max(season\_comfort\_zone_i)$ then
        $a_{h_i} = a_{h_i} - 1$
        $a_{c_i} = a_{c_i} - 1$
    if $IAT_i < min(season\_comfort\_zone_i)$ then
        $a_{h_i} = a_{h_i} + 1$
        $a_{c_i} = a_{c_i} + 1$
    $a_i = (a_{h_i}, a_{c_i})$
    if $current\_dt_i.weekday \geq 5$ or $current\_dt_i.hour$ in range(22,6) then
        $a_i = (18.33, 23.33)$(°C)
```

- **AWAC** [78]: Advantage Weighted Actor Critic (AWAC), it enables rapid learning of skills with a combination of prior demonstration data and online experience. The implicitly constrained actor-critic algorithm is able to both train offline and continue to improve with more experience.

- **REDQ** [148]: It combines the randomized ensemble Q-functions to improve sample efficiency along with a proportional-derivative (PD) controller to tune the hyperparameter of the weight of the behavioral cloning term $\alpha$ in Eq. 5.1 with a target score and current episodic return.

- **TD3+BC-FT**: From TD3+BC [33] to TD3 [35], we directly convert the offline TD3+BC to TD3 by removing the behavior cloning term shown in Eq. 5.1 at the beginning of the offline-to-online training, i.e., by setting $\alpha = 0$ in the fine-tuning stage.

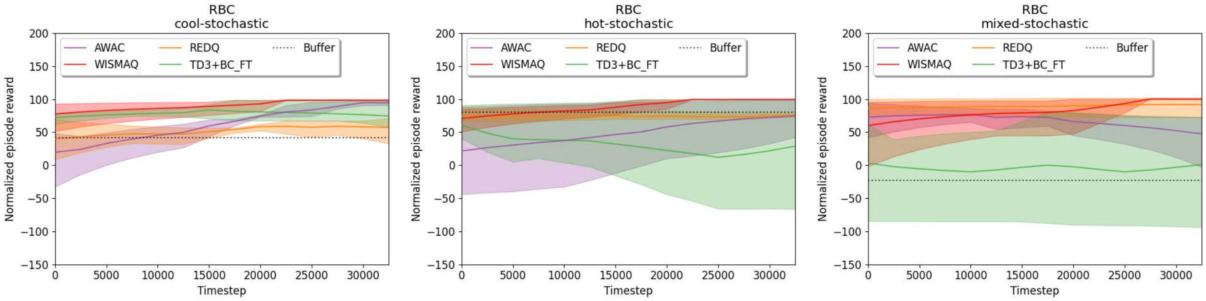


**Figure 5.4**: Learning curves of O2O models learn from RBC buffers

**Table 5.2**: Scores comparison between WISMAQ and other state-of-the-art method.

| Task/Algo. | AWAC | WISMAQ | REDQ | TD3+BC-FT |
|---|---|---|---|---|
| RBC-hot | 75.9±21.5 | **99.7±0.1** | 74.7±3.4 | 26.9±69.3 |
| RBC-mixed | 42.7±42.2 | **100.2±0.2** | 91.6±13.7 | 5.3±73.7 |
| RBC-cool | 94.6±2.5 | **98.6±0.1** | 58.3±20.6 | 72.6±22.1 |
| Sum | 213.2±66.2 | **298.5±0.4** | 224.6±37.7 | 104.8±165.1 |

In terms of optimization goals of energy consumption and thermal penalty. We normalize WISMAQś metrics as ones and compare other state-of-the-art methods in Fig. 5.5. As the figure 5.5 demonstrates, our method achieves a substantial improvement in comfort penalty (purple bars). Meanwhile, it maintains a comparable energy consumption among

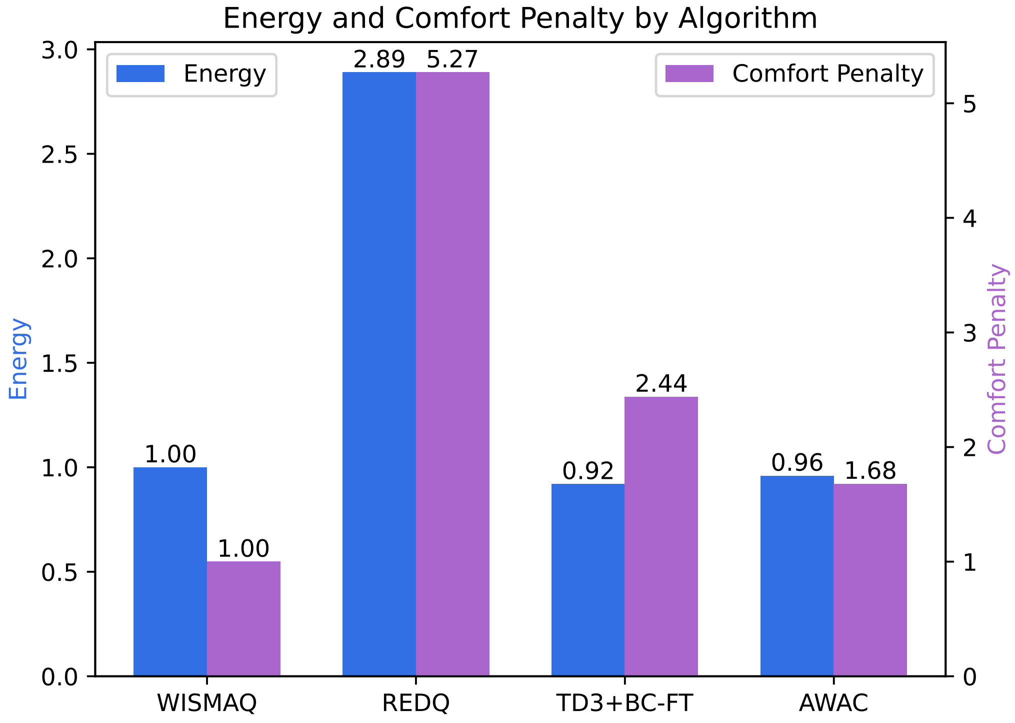


**Figure 5.5**: Compare the optimization objectives by algorithms, WISMAQś optimization objectives are normalized as 1, and the data shown is the summation of all six tasks across 3 different initialization conditions.

the state-of-the-art methods.

Together from Fig. 5.4 and Table 5.2[1], we can see that WISMAQ can maintain the pre-trained models' capacity. It answers the **Q.1** in Sec.5.1 And further improve them as training continues (corresponds to **Q.2**). Overall, WISMAQ has a 32.9% improvement from the next best state-of-the-art method AWAC. Also, WISMAQ shows a substantially more stable output with only 0.4% total standard deviation in scores of the last 5 evaluation episodes across six tasks. While other methods fail to output stable performance across different environment resets in the late training stages, WISMAQ could still learn policies with higher scores across different random seeds.

### 5.5.3 Data Efficiency Experiment

To answer the research question **Q.3** in Sec.5.1, we executed a series of data efficiency experiments to observe how the amount of data affects the training. We train with three different buffer sizes containing one year of data, four months, and one week:

- **Offline WISMAQ**: Pure offline training, RBC buffers are downsampled to the varied sizes as mentioned.
- **Online WISMAQ**: Pure online training, buffers initialized with zero transitions.
- **RBC**: Deploy RBC policy and evaluate.

[1] scores are normalized as the expert policy is 100 and the random policy is 0. The format is avg.±std. between 3 random initialization conditions of the final 5 evaluations. The highest score in a particular task is highlighted with bold font.

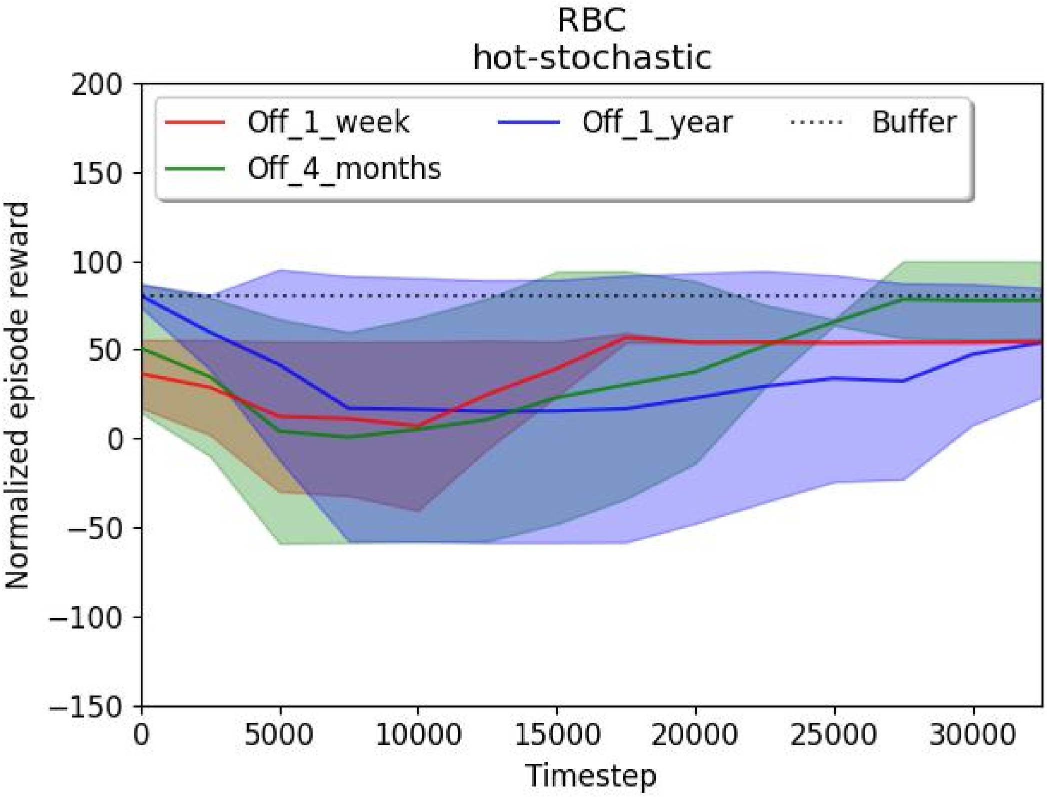


**Figure 5.6**: Offline training with varied max sizes of buffer.

Fig. 5.6 demonstrates that with pure offline WISMAQ training, we simulate the scenarios of different amounts of accessible data:{1 week, 4 months, and 1 year}. It is intuitive that with the smaller size of buffers, the agent would learn faster since as training continues, the better policies generate a higher quality of experience replay [28]. However, it comes with a less stable policy and could lead to catastrophic forgetting (1 week) for this. With too much data the model would learn from the old distribution which might damage the performance (1 year). Finding the optimized size of the buffer is also a crucial factor in off-policy learning.

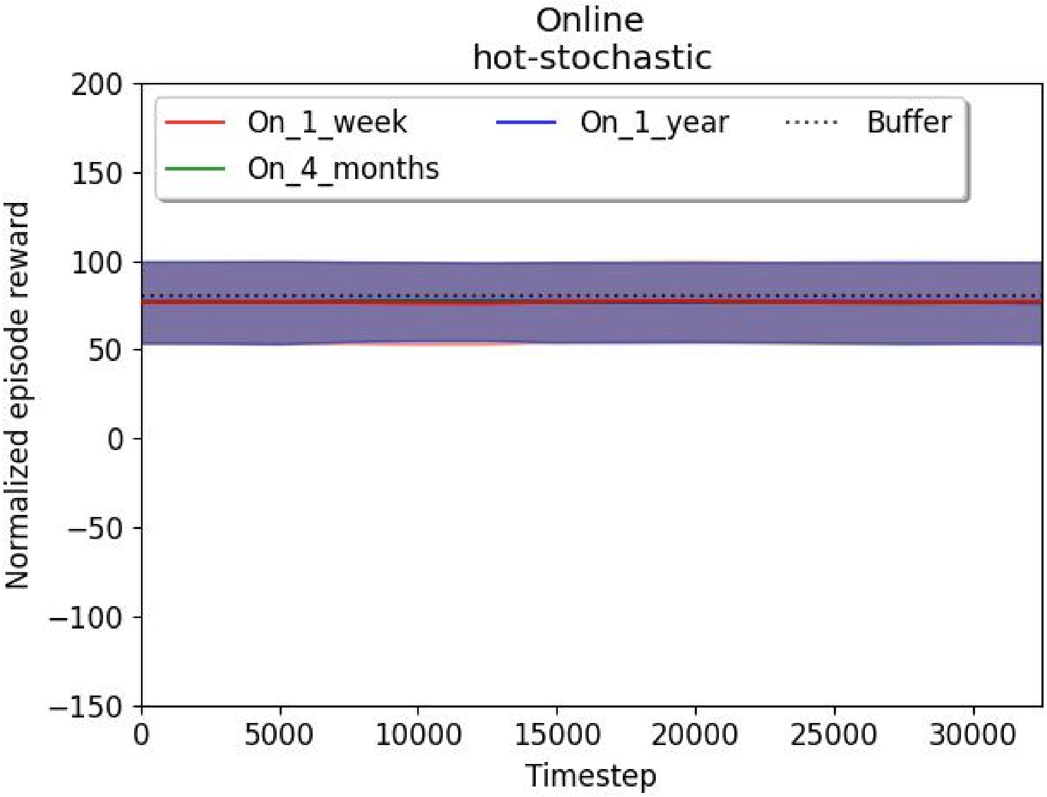


**Figure 5.7**: Online training with varied max sizes of buffer.

In Fig. 5.7 pure online WISMAQ are not able to learn a better policy with varied sizes

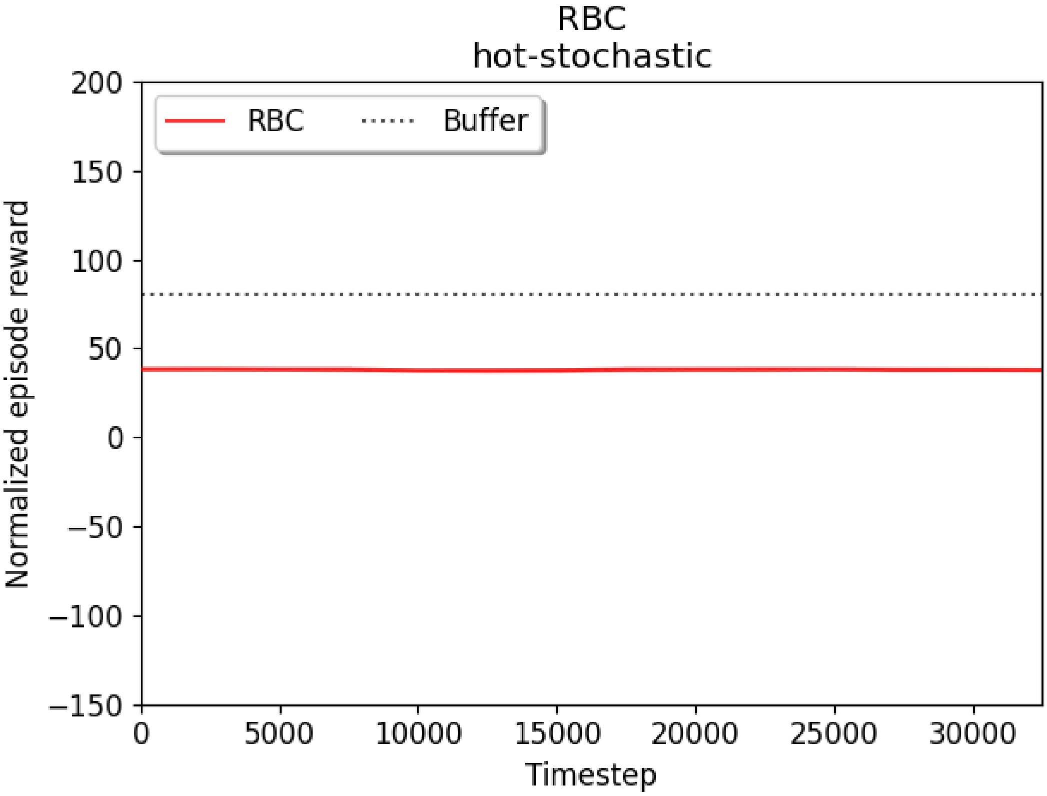


**Figure 5.8**: RBC (Rule-Based Control) deployment.

of the buffer (all three learning curves overlapped). Compared with our main experiments where agents were pre-trained with offline data they have a better knowledge of entire state-action spaces and their corresponding values. It indicates WISMAQ requires prior knowledge of the environment for a more promising outcome.

Fig. 5.8 is treated as a reference level to show that even with constant RBC policy, we cannot guarantee the RBC policy's performance with varied environment initializations.

### 5.5.4 Ablation Experiment

To demonstrate the capability of our add-on methods, we conducted a series of ablation experiments to validate their necessity (Fig. 5.9). We run WISMAQ "no_cer", and "no_WISMAQ" on the same task - cool weather with RBC buffers, the result indicates that without WISMAQ the model learns similarly to an offline model. Without CER, the model could adapt to the latest distribution drsift and finally fail to improve itself in the long run. Combining these methods altogether will boost the entire learning ability than separately.

### 5.5.5 Sensitivity Experiments

Deep Reinforcement Learning methods are known for their sensitivity to hyperparameter settings [3]. Thus, we have conducted a series of experiments to demonstrate how the hyperparameter settings affect the models' performances.

To experiment on the number of bootstrapping ensemble models, Fig. 5.10 indicates the higher number of ensemble models yield an overall faster convergence of the policy. We

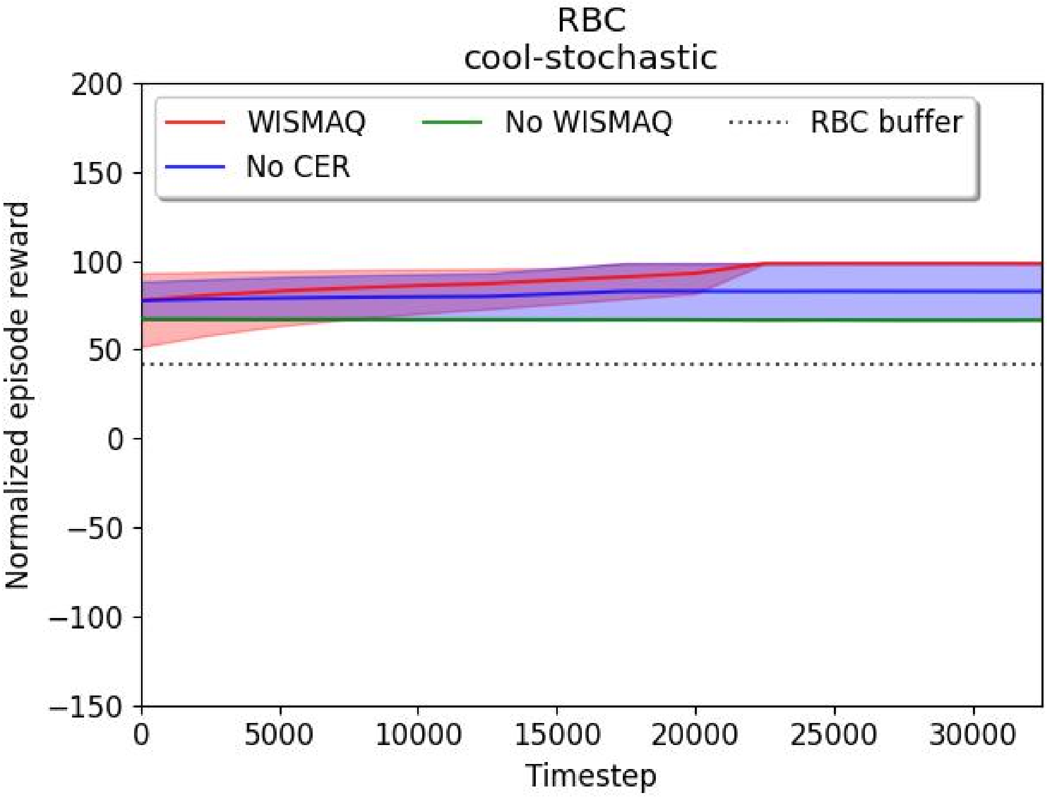


**Figure 5.9**: Ablation experiment.

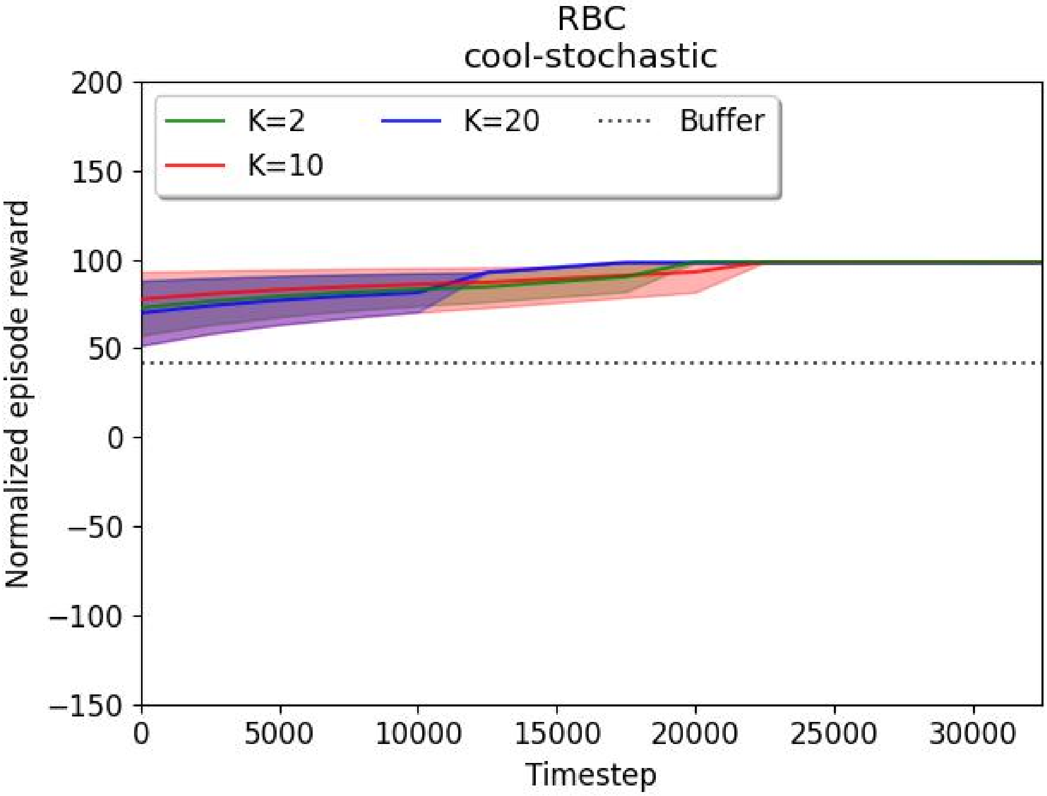


**Figure 5.10**: Sensitivity experiment on hyperparameter $K$, the number of ensemble models.

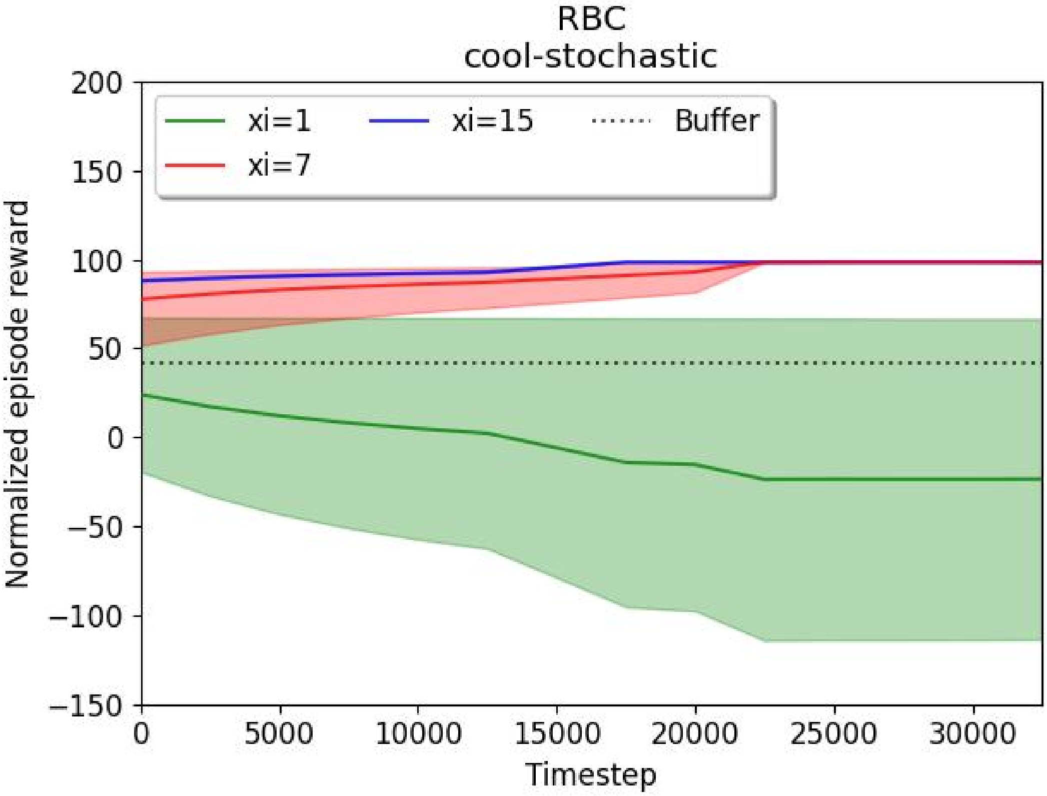


**Figure 5.11**: Sensitivity experiment on hyperparameter $\xi$, the weight of the WISMAQ loss term in actor training.

should consider the trade-off between computation resources and the convergence speed to decide the number of ensemble model $K$.

For the hyperparameter $\xi$ which weights the WISMAQ loss term in policy training, Fig. 5.11 shows with smaller weight $\xi = 1$, the model is unable to learn a better policy because the behavioral cloning term dominates. While a higher $\xi$ value leads to a more greedy fashion during policy learning, it might suffer from the inaccuracy of the value estimation. Thus, it is recommended to optimize this hyperparameter for each environment.

### 5.5.6 Adaptability Experiment

Furthermore, we want to examine the portability of our model when deployed to different environment settings. We conducted experiments with another environment - A datacenter. This environment contains 29 state space dimensions and 4 action space dimensions (for details please see Appendix.C.1). As the experimental result demonstrates, WISMAQ can learn a policy that is similar to expert policy while other methods could not adapt to the distribution drift as training goes on.

**Table 5.3**: Scores comparison of the scalability experiment.

| Datacenter | AWAC | WISMAQ | REDQ | TD3+BC_FT |
|---|---|---|---|---|
| RBC-hot | -39.5±2.0 | **100.0±0.0** | 13.7±37.9 | -78.0±3.7 |

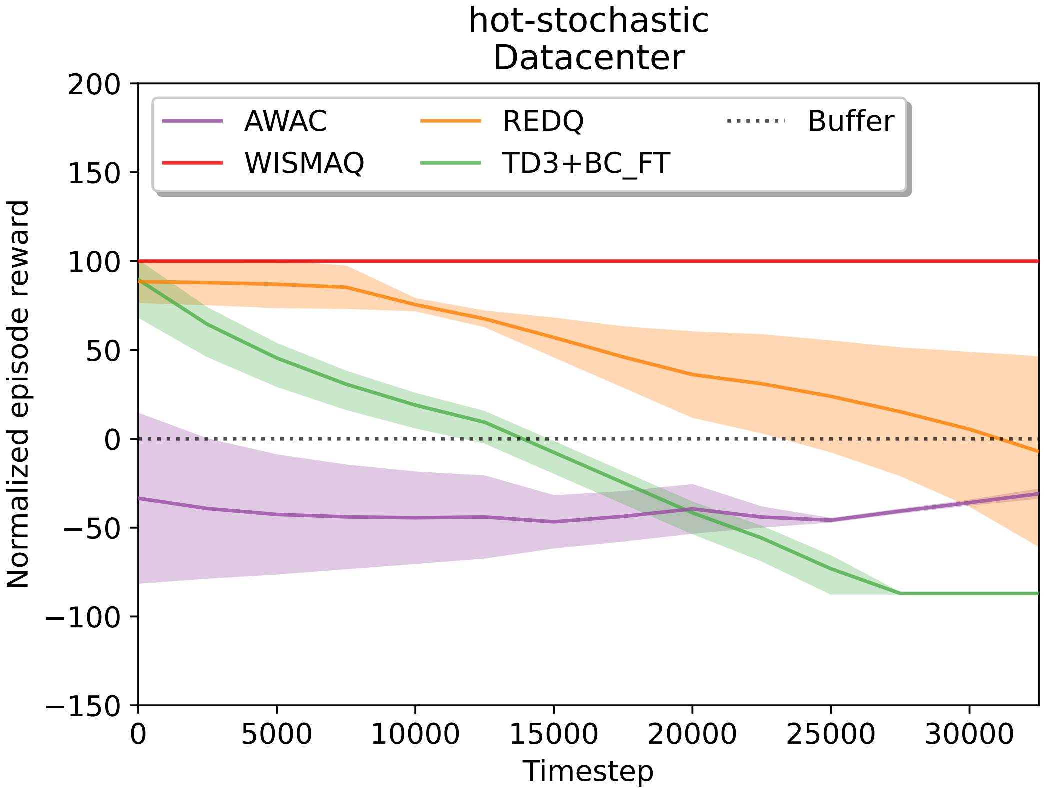


**Figure 5.12**: Scalability experiment with

data center environment.

## 5.6 Discussion and Conclusion

WISMAQ enables us to regularize the agent's policy in offline-to-online RL for use in HVAC control. It automatically tunes the actor loss that is designed to increase the average Q-value of the sampled batches in the experience replay. The use of a Simple Moving Average of the mean Q-value is key to our algorithm and could reflect the actual trend of the policy learning and its corresponding value estimation. Our experiments in the simulation environments indicate that WISMAQ not only maintains the pre-trained model capacity but also further learns a better policy as training goes on with RBC buffers.

The limitation of our method is as seen in higher dimensions of the state-action spaces. The effect of the curse of dimensionality increases which might lead to less accurate value estimation. Thus, the actual trend of the mean Q-value would be inherited with higher uncertainty. However, this issue could be mitigated by learning a lower bound of the value estimation or the help of ensemble Q-network for a more accurate prediction.

Our study encourages domain experts to explore the possibility of offline-to-online reinforcement learning applied in energy systems. Since a significant amount of data has been stored and should be utilized efficiently with data-driven methods.

Chapter 5, in part, is a reprint of the material that will be submitted in the future by the authors Hsin-Yu Liu, Bharathan Balaji, Rajesh Gupta, and Dezhi Hong. with a tentative title: "Policy Regular- ization for Offline-to-Online Reinforcement Learning in HVAC Control". The dissertation author is the primary investigator and author of this paper.

# Chapter 6

# Future Work and Conclusion

## 6.1 Future Work

The exploration of Offline Reinforcement Learning (Offline RL) and its transition to Offline-to-Online RL paradigms opens up a new era in building control systems, significantly enhancing their efficiency and adaptability. While these advancements have set a robust foundation, the evolving landscape of smart buildings and the increasing emphasis on sustainability and occupant comfort beckon further innovation. This chapter outlines potential future directions in leveraging Reinforcement Learning to address the emerging challenges and opportunities in building control systems.

**Adaptive and Generalizable RL Frameworks** We need RL frameworks that exhibit high levels of adaptability and generalization across diverse building architectures and environmental conditions. This entails creating algorithms that can efficiently transfer learned policies from one building to another, reducing the need for extensive retraining. Techniques such as meta-learning and transfer learning could play pivotal roles in achieving this goal, enabling RL agents to apply knowledge gained in one context to novel situations.

**Integration of Multi-Agent Systems** As buildings evolve into more complex systems of cyber-coupled subsystems, the implementation of multi-agent RL (MARL) systems offers a promising avenue for managing the interactions between various subsystems, such as HVAC, lighting, and security. MARL can facilitate cooperative strategies that optimize overall energy consumption and occupant comfort more effectively than isolated agents. Future work will need to address the challenges of coordination and communication between agents to ensure cohesive operation.

**Exploiting Deep Generative Models** The incorporation of deep generative models within RL frameworks can enhance the prediction and simulation capabilities essential for effective building control. By generating realistic scenarios of environmental conditions and occupant behavior, these models can provide RL agents with rich, synthetic datasets for training, improving their decision-making under uncertainty. Investigating the synergy between generative models and RL could unlock new potential for predictive building management.

**Human-in-the-Loop RL Approaches** Integrating human feedback directly into the RL loop represents a significant frontier. Human-in-the-loop approaches can ensure that the RL policies not only optimize energy efficiency but also align with human comfort and preferences. We need mechanisms for efficiently incorporating subjective feedback and adapting RL policies accordingly, potentially through preference-based or inverse RL techniques.

**Addressing Safety and Reliability** As RL-based control systems become more prevalent, ensuring their safety and reliability, especially in critical systems like HVAC, is paramount. Future work should focus on developing safe RL algorithms that incorporate risk assessment and management directly into their learning processes. This includes exploring safe exploration strategies and robust policy evaluation methods that can guarantee the safety of actions before their deployment in the real world.

**Leveraging Edge Computing** The integration of edge computing with RL can significantly enhance the responsiveness and efficiency of building control systems by processing data closer to the source. This approach minimizes latency and reliance on cloud services, enabling real-time adjustments to building controls. Investigating RL algorithms optimized for edge devices, considering their computational constraints, will be crucial for advancing smart building technologies.

## 6.2 Conclusion

This dissertation covers studies of policy regularization in model-free building control via Offline, Online, and Offline-to-Online RL. Provides robustness and stability while maintaining scalability and learnability during the agent training.

The journey from Offline RL to its integration into online environments has marked a significant advancement in building control strategies. However, the continuous evolution of smart building technologies and the growing demand for sustainable, comfortable, and energy-efficient living spaces present new challenges and opportunities for RL research. By addressing these future directions, we can further harness the power of RL to revolutionize building control systems, making them more adaptive, efficient, and aligned with human needs.

This exploration not only contributes to the field of building automation but also sets a precedent for applying advanced RL techniques in other domains where efficiency and adaptability are paramount.

# Appendix A

# Safe HVAC Control via Batch Reinforcement Learning

## A.1 Munchausen Regularizaion

### A.1.1 Motivation

Current batch/offline RL methods are mainly focused on utilizing statistical methods or using regularization methods to mitigate the effect of distribution drift. Pessimistic Q-Learning (PQL [70]) adds a state Variational Auto Encoder (VAE [51]) and uses a filtration function to avoid Q-update when state-action visitation is not frequent enough in the batch. Bootstrapping Error Accumulation Reduction (BEAR [58]) uses the sampled version of Maximum Mean Discrepancy (MMD) between the unknown behaviour policy and the actor as constraint to avoid actions that lie outside of the training data distribution.

Batch Constrained Q-learning (BCQ [36]) also minimizes the distance of selected action to the data in the batch and leads to states where familiar data could be observed. While effective, however, none of them considers another source of learning—the current policy.

Model-free batch RL is challenging because it is in the deadly triad of off-policy learning, function approximation, and bootstrapping [110]. The key insight of our work is that we improve offline methods in the off-policy Q-update itself. While other works focus on the extrapolation errors, bootstrapping errors, and function approximations.

### A.1.2 Methodology

We add a scaled log-policy term in the Q-update step in the Batch RL Q-network architecture inspired by Munchausen-RL [124]. State-of-the-art batch RL algorithms, such as PQL and BEAR, are based on BCQ's architecture, and BCQ uses double-clipped Q-learning architecture. We follow a similar methodology and modify the Q-update step from:

$$r + \gamma \max_{a_i} \left[ \lambda \min_{j=1,2} Q_{\theta'_j}(s^{'}, a_i) + (1-\lambda) \max_{j=1,2} Q_{\theta'_j}(s^{'}, a_i) \right] \text{ to}$$

$$r + \alpha_m \tau_m \ln(softmax(\tfrac{Q_{\theta'}}{\tau_m})) + \gamma \max_{a_i} \left[ \lambda \min_{j=1,2} Q_{\theta'_j}(s^{'}, a_i) + (1-\lambda) \max_{j=1,2} Q_{\theta'_j}(s^{'}, a_i) \right]$$

with the Munchausen term highlighted in red, where $\alpha_m$ and $\tau$ are hyperparameters[1]. Algorithm 7 gives the full description. Additionally, we also adapt Prioritized Experience

[1] M-RL regularization consists of two parts: the first part is the one we add on BRL architectures by using Kullback-Leiber divergence to penalize policies that are far from the previous policy, and the other is using an entropy term to penalize policies that deviate far from uniform distribution [121]. Our evaluation shows that penalizing only the first term yields the best outcome.

Replay (PER [97]) with BCQ. We compute the rank-based probability $P(j)$ based on priority $p_j^\alpha$, importance-sampling weight $\omega_j$, and TD-error $\delta_j$. For each mini batch $k$ for $j = 1$ to $k$:

$$P(j) = p_j^\alpha / \sum_i p_i^\alpha$$

$$\omega_j = (N \cdot P(j))^{-\beta} / \max_i \omega_i$$

$$\delta_j = R_j + \gamma_j Q_{target}(S_j, argmax_a Q(S_j, a)) - Q(S_{j-1}, A_{j-1})$$

Where $N$ is the size of the replay period. We update the transition priority $p_j \leftarrow |\delta_j|$. $\alpha$ and $\beta$ are the exponent hyperparameters. Finally, we update the critic network with:

$$\theta_i \leftarrow argmin_{\theta_i} N^{-1} \sum \omega(y - Q_{\theta_i})^2$$

We evaluate against the state-of-the-art BRL methods: BCQ, PQL, and BEAR, and compare their modified versions with Munchausen and PER variants. [2]

**Table A.1**: Evaluated Algorithm Variants

| Name | Description |
|---|---|
| BCM | BCQ with Munchausen-RL |
| PML | PQL with Munchausen-RL |
| BCQ_PER | BCQ with PER |

[2] We also implement PQL_PER, however due to the heuristic in PQL that avoids Q-update when visiting low-data region, the results are not improving, so we omit it in the comparison.

```
Algorithm 7: BCM algorithm
Input : Batch $\mathcal{B}$, horizon $T$, target network update rate $\tau$, mini-batch size $N$,
        max perturbation $\Phi$, number of sampled actions $n$, minimum weighting
        $\lambda$, M-RL hyperparameters $\alpha_m$ and temperature parameter scaling the
        entropy $\tau_m$
Initialize Q-networks $Q_{\theta_1}, Q_{\theta_2}$, perturbation network $\xi_\phi$, and VAE
  $G_\omega = \{E_{\omega_1}, D_{\omega_2}\}$, with random parameters $\theta_1, \theta_2, \phi, \omega$, and target networks
  $Q_{\theta_1'}, Q_{\theta_2'}, \xi_{\phi'}$ with $\theta_1^{'} \leftarrow \theta_1, \theta_2^{'} \leftarrow \theta_2, \phi^{'} \leftarrow \phi$
for $t \leftarrow 1$ to $T$ do
    Sample mini-batch of $N$ transitions $(s, a, r, s^{'})$ from $\mathcal{B}$
    $\mu, \sigma = E_{\omega_1}(s, a), \widetilde{a} = D_{\omega_2}(s, z), z \sim \mathcal{N}(\mu, \sigma)$
    $\omega \leftarrow argmin_\omega \sum (a - \widetilde{a})^2 + D_{KL}(\mathcal{N}(\mu, \sigma) || \mathcal{N}(0, 1))$
    Sample $n$ actions: $\{a_i \sim G_\omega(s^{'})\}_{i=1}^n$
    Perturb each action: $\{a_i = a_i + \xi_\phi(s^{'}, a_i, \Phi)\}_{i=1}^n$
    Set value target:
    $y = r + \alpha_m \tau_m \ln(softmax(\frac{Q_{\theta'}}{\tau_m})) + \gamma \max_{a_i} \left[\lambda \min_{j=1,2} Q_{\theta_j'}(s^{'}, a_i) + (1 - \lambda) \max_{j=1,2} Q_{\theta_j'}(s^{'}, a_i)\right]$
    $\theta \leftarrow argmin_\theta \sum (y - Q_\theta(s, a))^2$
    $\phi \leftarrow argmin_\phi \sum Q_{\theta_1}(s, a + \xi_\phi(s, a, \Phi)), a \sim G_\omega(s)$
    Update target networks: $\theta_i^{'} \leftarrow \tau\theta + (1 - \tau)\theta_i^{'}$
    $\phi^{'} \leftarrow \tau\phi + (1 - \tau)\phi^{'}$
```

### A.1.3 Experimental Setup and Result

#### Experimental Setup

We evaluate our methods on MuJoCo [113] environments similar to prior works but use the latest version: *Hopper-v3*, *HalfCheetah-v3*, and *Walker2d-v3*. We use Deep Deterministic Policy Gradient (DDPG [66]) to generate buffers after training for one million time steps with a $\mathcal{N}(0, 0.1)$ Gaussian noise to select random actions. Then the agent is used to generate buffers across five random seeds also with a $\mathcal{N}(0, 0.1)$ Gaussian noise to emulate stochastic processes.

All of our experiments are conducted with Intel Xeon Gold 6230 CPUs (2.10GHz) and NVidia Quadro RTX 8000 GPUs with Ubuntu 18.04 OS. All results shown are trained and evaluated with five buffers with different random seeds.

#### Metrics and Results

We report the mean and median scrores across our experiments. Following Agarwal et al. [2], we also report inter-quantile mean (IQM), optimality gap, performance profile and probability of improvement to account for inherent uncertainty in deep RL training.

**Aggregate Metrics** In Fig.A.1, aggregate metrics are with 95% of confidence interval (CI) and stratified sampling using percentile bootstrapping $50K$ times. IQM discards the bottom and the top 25% of the scores, then calculates the mean. Optimality gap is the amount by which the algorithm fails to meet a minimum score of $\gamma = 1.0$, typically we set the aim as the normalized human/expert score. We can see that PML has a smaller optimality gap and higher median, IQM, and mean compared with the second-best algorithm, PQL.

**Performance Profile** Performance profile is commonly used in benchmarking optimization software. However, it does not consider uncertainty estimation. A revised version of performance profile is called run-score distribution. It shows the fraction of runs above a certain normalized score. It is an unbiased estimator of the underlying distribution and more robust than average-score distribution. In Fig. A.2 we observe that PML outperforms other methods almost under any condition. On the other hand, the addition of Munchausen regularization and PER are helpful for improving BCQ. The results shown here are bootstrapped with $2K$ times.

**Probability of Improvement** Probability of improvement is a metric which indicates how likely one method outperforms the other on a randomly selected task. This metric does not account for the size of improvement. As we can see in Fig. A.3, PML is most likely to dominate among the methods we have evaluated.The results shown here are bootstrapped with 200 times.

**Learning Curves** Fig. A.4, A.5, and A.6 illustrate the learning curves of all the algorithms evaluated with training time steps as the x-axis and the average episode rewards on the y-axis. Each solid line shows the average between runs, and half-transparent regions indicate the range. The results again verify the robustness of the add-on of Munchausen regularization.[3]

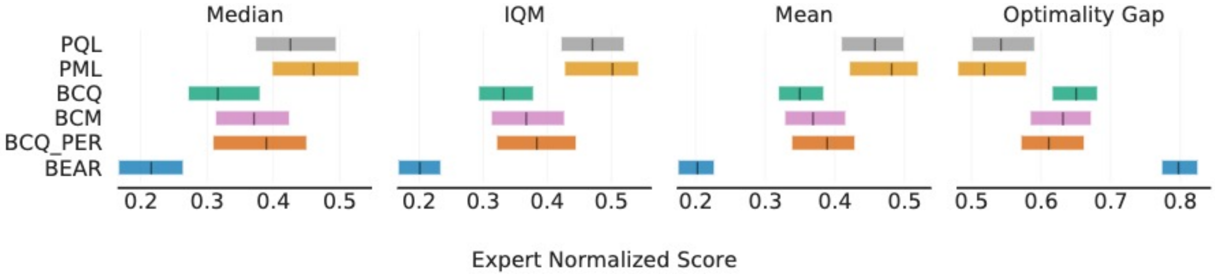


**Figure A.1**: Aggregate Metrics

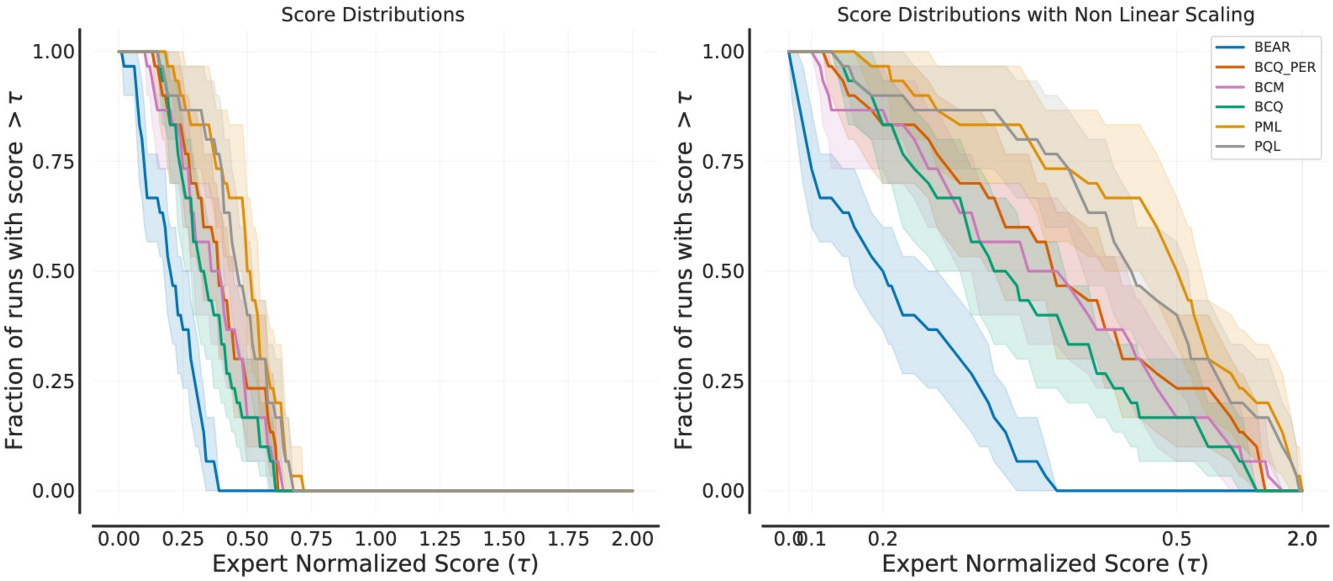


**Figure A.2**: Score distribution with linear/non-linear scaling

[3] All results are based on five runs, except for BEAR, some runs were aborted due to MuJoCo simulator feedbacks system state for large numbers or inf./NaN.

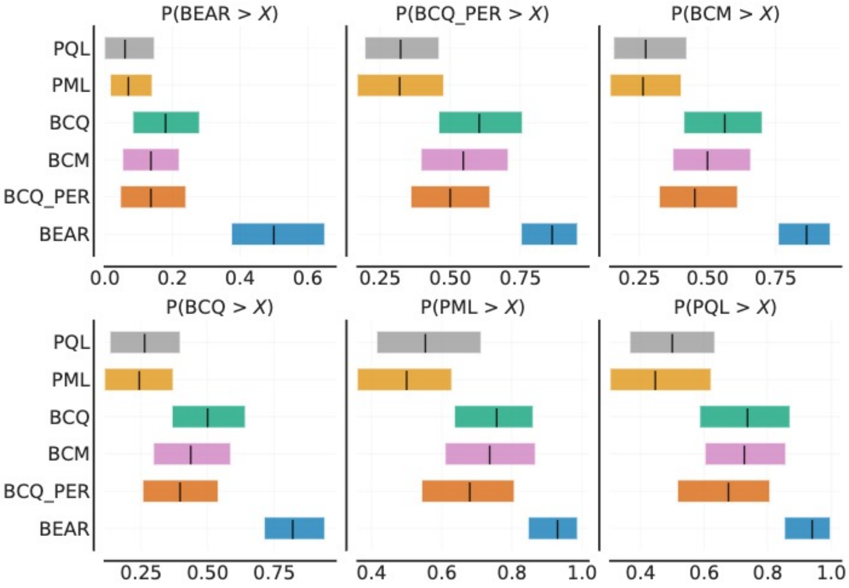


**Figure A.3**: Probabilities of improvement

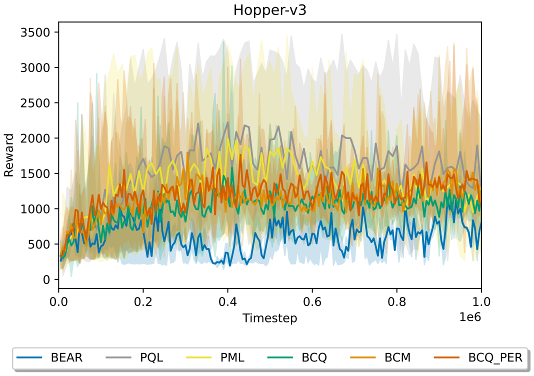


**Figure A.4**: Learning curves of *Hopper-v3*

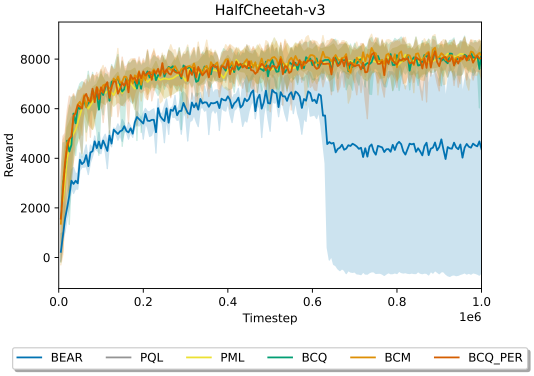


**Figure A.5**: Learning curves of *HalfCheetah-v3*

### A.1.4 Conclusion and Discussion

In this work, we show that Munchausen regularization is effective in improving BRL methods. It penalizes policies that are far from the previous ones. It can serve as a

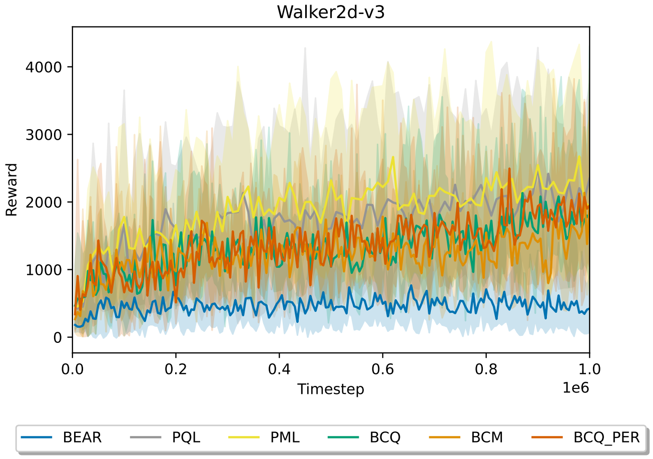


**Figure A.6**: Learning curves of *Walker2d-v3*

strong learning signal to enhance the performance of models. Moreover, prioritized replay with weighted importance sampling could also improve BRL methods with consistent Q-update. Due to the massive amount of resources required in continuous spaces DRL algorithm evaluation, usually DRL studies conduct a handful of runs ($3 \sim 10$). We use aggregate statistical metrics that consider the uncertainty to provide a more robust comparison. These results are encouraging to us to discover more opportunities to boost BRL performances with regularization approaches. We expect to implement more benchmarks and further improvements as our future work.

### A.1.5 Experiments Details

#### Parameters

For researchers to better reproduce our results, we provide the hyperparameters used in our experiments. For most of the models, we follow their default settings unless otherwise recommended. We do not fine-tune the hyperparameters of the BRL algorithms and use the reported values in the literature [36, 70, 58], and we keep the architecture of actor-critic networks for a fair comparison. Modifying the architecture or any detail of implementation might lead to a large difference in performances. [42] In PQL, we scale the maximum state VAE training steps according to the ratio of PQL's MuJoCo buffer size to our building buffer size. For all the network architectures we follow the original setups. The details of the hyperparameters are listed in Table A.2.

### Data Monitored

In the evaluation processes, we monitored all the states as time series to observe if there is any abnormality. Also, to inspect how BRL methods optimize the target objectives.

As shown in Fig. A.7, it is an example of how the thermal comforts of historical weeks vary under rule-based control. Apparent periodic patterns are observed which follow the OAT trends during the week. It indicates that RBC cannot compensate the OAT variations as BRL method (Fig. 2.8).

In Fig. A.8, it shows the time series of the states observed during BRL evaluation. Our BRL method BCM keep zone air temperature setpoint (ZNT StPt) in a narrow range stably, thus, keep the zone air temperature readings (ZNT) in a reasonable range to maintain thermal comforts while no constraints are applied, which is different from online RL methods where the range of actions are constrained by human experts as hard rules.

**Table A.2**: Hyperparameter Settings of evaluated methods

| | BCM | BCQ | BEAR | PQL |
|---|---|---|---|---|
| $\gamma$ | 0.99 | 0.99 | 0.99 | 0.99 |
| $N$ | 100 | 100 | 100 | 100 |
| $\tau$ | 0.005 | 0.005 | 0.005 | 0.005 |
| $\lambda$ | 0.75 | 0.75 | 0.75 | 0.75 |
| $\Phi$ | 0.05 | 0.05 | – | 0.1 |
| $\alpha_m$ | 0.9 | – | – | – |
| $\tau_m$ | 0.03 | – | – | – |
| clip value min. | -1 | – | – | – |
| backup | – | – | – | Q-max |
| QL noise | – | – | – | 0.15 |
| $b$ percentile | – | – | – | 2 |
| max state VAE trainstep | – | – | – | $2e4$ |
| Policy update version | – | – | 0 | – |
| MMD matching # samples | – | – | 5 | – |
| MMD sigma | – | – | 20 | – |
| Kernel type | – | – | Laplacian | – |
| Lagrange threshold | – | – | 10 | – |
| Distance type | – | – | MMD | – |

$\gamma$: discount factor, $N$: mini-batch size, $\tau$: target network update rate, $\lambda$: minimum weighting between two Q-networks, $\Phi$: max perturbation on action, $\alpha_m$: Munchausen scaling term, $\tau_m$: entropy temperature, clip value min.: minimum clipping value on Munchausen term

### A.1.6 Experiment with safe minimum airflow

#### Motivation

Indoor environment and indoor gatherings present a disease spreading risk as virus-laden aerosol lingers in indoor air for hours at high concentrations [64] rather than being

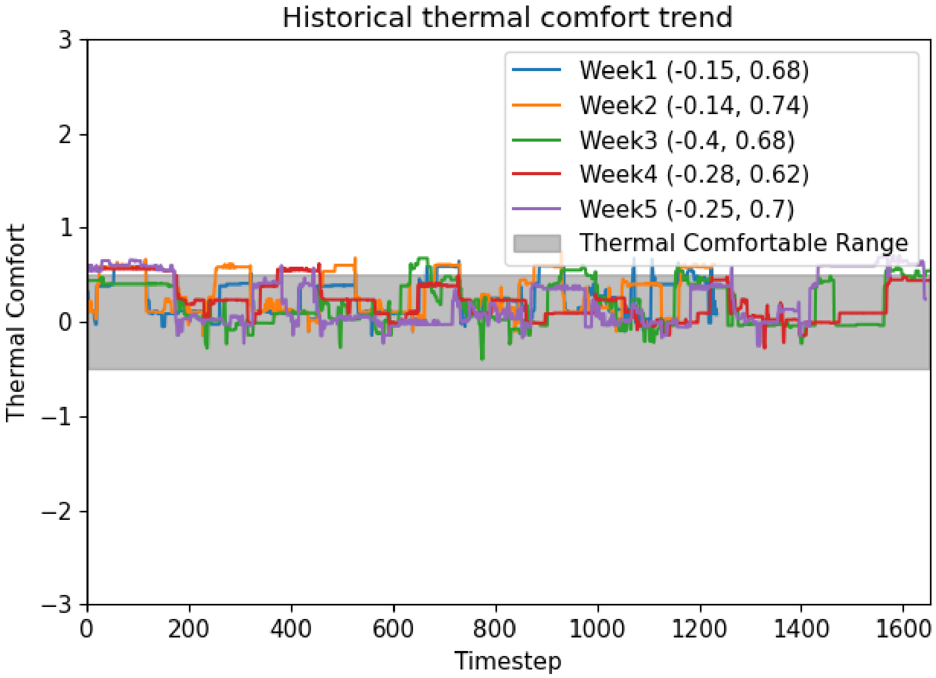


**Figure A.7**: An example of historical thermal comfort trends in top-5 similar OAT weeks

quickly dispersed and destroyed through UV (sun)light outdoors. Accumulated exposure to viral load over time is an important risk determinant for an individual to be infected [19]. In the context of the current pandemic caused by the spreading of the SARS-CoV-2 virus that causes COVID-19 disease, many efforts are underway to control its spread for the public healthcare system to maintain its capacity and reduce fatalities. We believe that a well-designed operation of the HVAC system can be a critical means to reduce the likelihood of spreading events by appropriately directing airflows. HVAC societies such as ASHRAE and REHVA have recommended high rates of air circulation and an increased fraction of fresh air. This is typically measured by air changes per hour, or ACH, in a given enclosed space or the entire building. ACH is computed by the air volume added

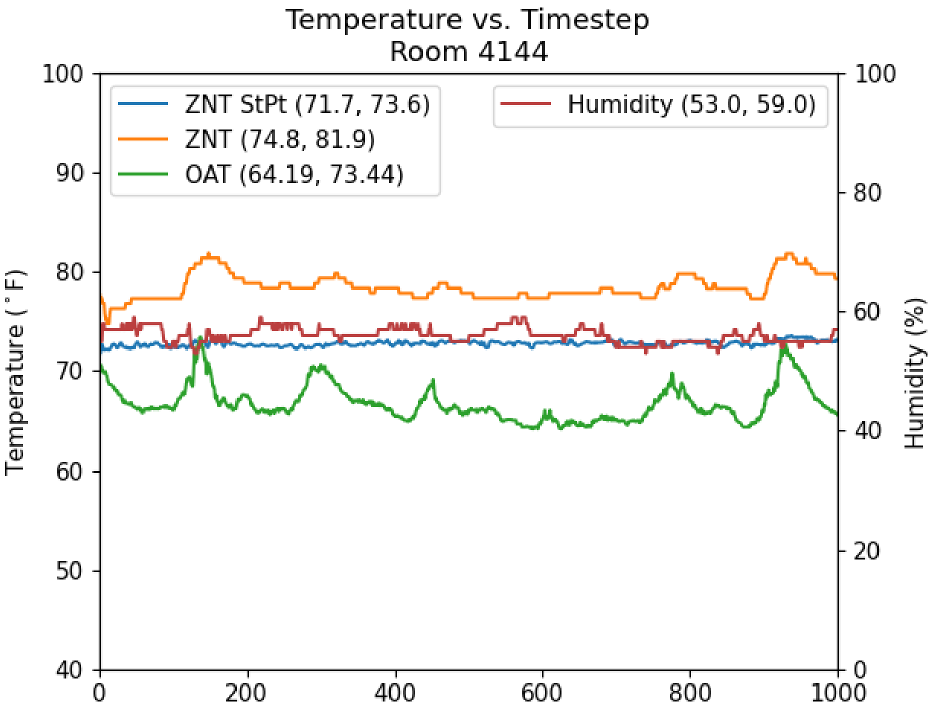


**Figure A.8**: States in BCM evaluation week

to or removed from space in an hour divided by the total volume of the space [111]. For air impurities removed by fresh air, unit ACH is then a time constant that represents the rate of dilution in infectious particles caused by the introduction of fresh-air [19]. ACH is increased primarily by increasing the ratio of fresh air and the speed of airflow supplied to a given space. Typically, commercial buildings are designed to achieve ACH levels of 3-5 whereas more sensitive areas in hospital settings could be as high as 12 ACH [16]. Achieving a substantially high ACH level in a typical office building is challenging due to the cooling capacity of the equipment [32], and thus in our study, we seek to fulfill a minimum safe airflow requirement.

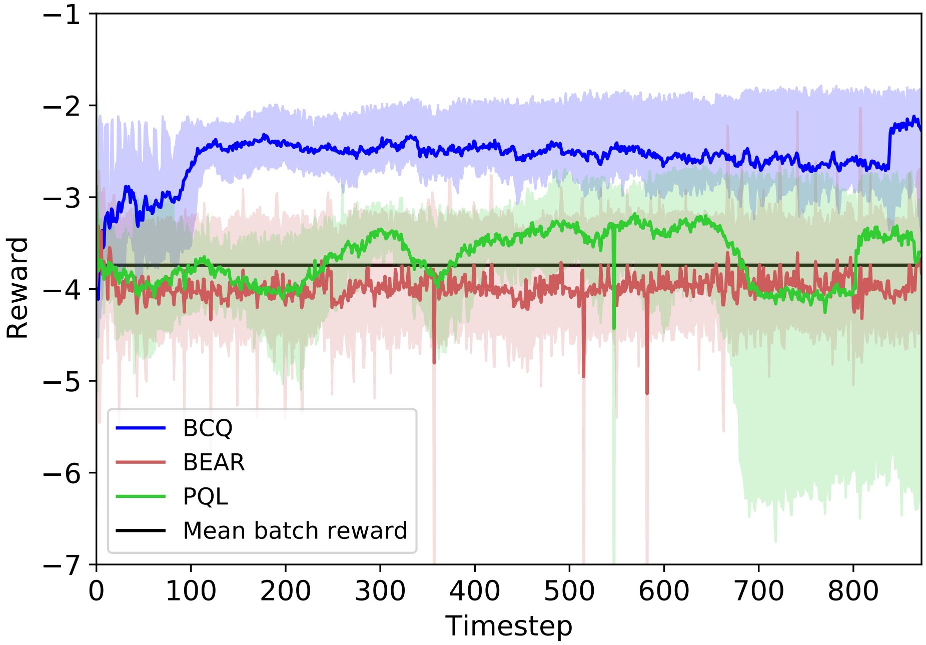


**Figure A.9**: Reward comparison (considering safe airflow)

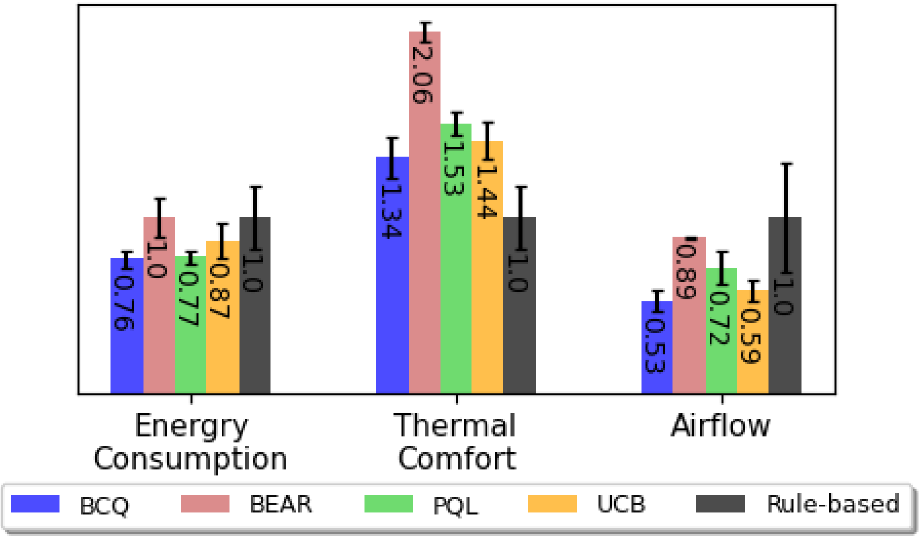


**Figure A.10**: Energy, thermal comfort, and airflow comparison

### Safe Airflow Level Guidelines

Various guidelines have been issued by ASHRAE[4], CDC[5], and the European union REHVA[6] on building operation to lower the risk of getting infected by the respiratory

[4] https://tinyurl.com/yy8f5faq

disease of the occupants through the air during the COVID-19 pandemic. These guidelines provide detailed recommendations regarding multiple aspects of building operation and share much in common, including, but not limited to, use of high-rating minimum efficiency reporting value (MERV) filters and/or UV-C lighting to treat the return air, 24/7 HVAC operation, no use of recirculated air (i.e. use 100% outside air), increased air change (ACH) rate during occupancy.

While comprehensive, these recommendations are difficult to implement altogether, if not completely impossible. The effects of these measures and their implications on the building systems with respect to energy consumption and occupants' thermal comfort still largely remain unclear to practitioners and residents. In our work, we maintain a safe airflow level in the zones we evaluate by requiring a minimum of 21.19 CFM per person (10L/s per person) [88] airflow in a space, which satisfies ASHRAE's, REHVA's, and CDC's requirements.

### Experiment Results

In Fig. A.9, we compare several state-of-the-art BRL methods as we did in our main experiments. The minimum safe airflow is calculated with the people occupied in the room, where we assume full occupancy.

The state, action, environment setups are all the same as our main experiments. Except for the reward function at time step $t$ is calculated with the following equation:

[5] `https://tinyurl.com/y9lczbwp`

[6] `https://tinyurl.com/yy8nzlmj`

$$R_t = -\alpha ReLU(|TC_t| - TC_c) - \beta s_t^{Sup} - \delta ReLU(A_{min}^{safe} - s_t^{Sup}), \tag{A.1}$$

In Eq.(A.1), $\alpha$, $\beta$, $\delta$ are the weights balancing between different objectives and could be tuned to meet specific goals, $TC_t$ is the thermal comfort index at time $t$, $TC_c$ is the requirement on thermal comfort, 0.5, and $s_t^{Sup}$ is the supply airflow at the time $t$, and we assume each room is fully occupied, leading to a constant $A_{min}^{safe}$ for each room based on the ACH requirement and number of people at full occupancy. The $ReLU$ (Rectified Linear Unit) activation function is used here to penalize any thermal comfort index that is out of the comfortable range and any airflow value that is lower than minimum safe airflow.

The results are run with two stacks of rooms per algorithm. And each stack of runs lasts approximately a week. The experiment result motivates us to improve from BCQ, since it outperforms the others in the real HVAC environments. The buffer is the same as our main experiments with an entire year of records. And the evaluation time period is from June 1st to June 14th, 2021.

To further analyze the improvements of the target objectives, respectively. Fig. A.10 shows the comparison of energy consumption, thermal comfort, and airflow readings. In this figure, RBC value of each category is normalized as one. We could observe that in summer OAT weeks, BRL methods could save more energy compared with the results of our main experiments where evaluation is done in the Fall. BCQ is with a 24 percent of energy reduction cf. RBC due to a more efficient policy control with a more stable airflow and thermal comfort, as the error bars shown in the figure.

Appendix A.1, in part, is a reprint of the material that appears in the Offline Reinforcement Learning Workshop at Neural Information Processing Systems (NeurIPS Offline-RL Work- shop 2021), vol. 2021, 2021. By author Hsin-Yu Liu, Bharathan Balaji, Rajesh Gupta, and Dezhi Hong with the title - “Offline reinforcement learning with Munchausen regularization.” The dissertation author is the primary investigator and author of this paper.

# Appendix B

# Incorporating existing policies with Reinforcement Learning

## B.1 Experiment details

- **Software**
    - **Python**: 3.9.12
    - **Pytorch**: 1.12.1+cu113 [82]
    - **Sinergym**: 1.9.5 [47]
    - **Gym**: 0.21.0 [14]
    - **Numpy**: 1.23.1 [117]
    - **CUDA**: 11.2
- **Hardware**

- **CPU**: Intel Xeon Gold 6230 (2.10 GHz)
  - **GPU**: NVidia RTX A6000

- **Average training/evaluation time for RUBICON**: 4 hours 45 minutes 18 seconds

- **Benchmark implementations**
  - **DDPG**: We adopt the DDPG implementation in TD3 author-provided implementation
  - **TD3**: Author-provided implementation
  - **SAC**: We adopt CleanRL [45] implementation due to software version conflict with author-provided repository
  - **TD3+BC**: Author-provided implementation
  - **CQL**: We adopt d3rlpy [101] implementation due to software version conflict with the author-provided repository
  - **BCQ**: Author-provided implementation

## B.2 Environments, Learning curves, detailed scores, and additional experiments

- **Sinergym Environments** A single-story building divided into 5 zones (1 indoor and 4 outdoor). Its surface area is $463.6m^2$, and it is equipped with a VAV package

(DX cooling coil and gas heating coils) with fully auto-sized input as the HVAC system to be controlled.

- **BRL learning** We illustrate the learning curves of BRL methods learn from different quality of buffers for better visualization of comparison in Fig. B.1, B.2 and B.3
- **Behavioral agents** The behavioral agents' learning curves are demonstrated in Fig. B.4
- **CQL+RUBICON** Since CQL demonstrates better performance compared to other methods except for RUBICON (see Table 4.1), we conduct experiments combining CQL and our RUBICON method to learn from random and medium buffers since the performance of CQL-expert is already extremely well. The results in Figure B.5, B.6 and Table B.4, B.5 indicate that RUBICON also improves CQL. However, it does not consistently improve CQL's performance from task to task, which is not as we observe from TD3+BC to RUBICON learn from random/medium buffers (see Figure B.3, 4.3). Also, the improvement is limited and does not even reach the RBC policy performance, thus we did not continue exploring the possibility of combining CQL with RUBICON.
- **Learn from worsened RBC policies** We run another ablation experiment to observe how the quality of RBC policy affects the performance compared with RUBICON and baseline TD3+BC. We design two worsened RBC policies: The first one is a biased RBC where we modify the change in setpoints ($a_{h_i}$ and $a_{c_i}$) from 1 to 5 in Algo. 6, we name this method "RBC_CB" in Figure B.7. The other is to

replace RBC with random policy, it is named as "RBC_Random". From the results in Table B.7 we could find that even with constantly worsened RBC policy it still improves from baseline, However, it is still too aggressive for the models to learn a robust policy. And with random policy as a worsened RBC it is almost equivalent as no reference policy, the performance is similar to our baseline TD3+BC.

- **Non-selective experiments** In this experiment, we remove the dynamically weighted regularization. Instead, we regularize the behavioral policy and RBC policy simultaneously in every iteration of training (see Eq. B.1). The experimental results are shown in Table B.8. We observe that regularizing both policies at the same time deteriorates the model performance cf. RUBICON. Since in each iteration, one of RBC policy $\pi_{rbc}(s)$ and behavioral policy $\pi_b(s)$ yields a better action selection compared to the other. It emphasizes the necessity of dynamic weighting in the policy update steps.

$$\pi = \underset{\pi}{\operatorname{argmax}}\, \mathbb{E}_{(s,a)\sim\mathcal{D}}\left[\lambda Q(s,\pi(s)) - (\pi(s)-a)^2 - (\pi(s)-\pi_{rbc}(s))^2\right] \tag{B.1}$$

All learning curves are normalized with random policy as 0 and expert policy as 100, averaged with 3 random seeds and the scores shown in tables are the average and standard deviation last 5 evaluations.

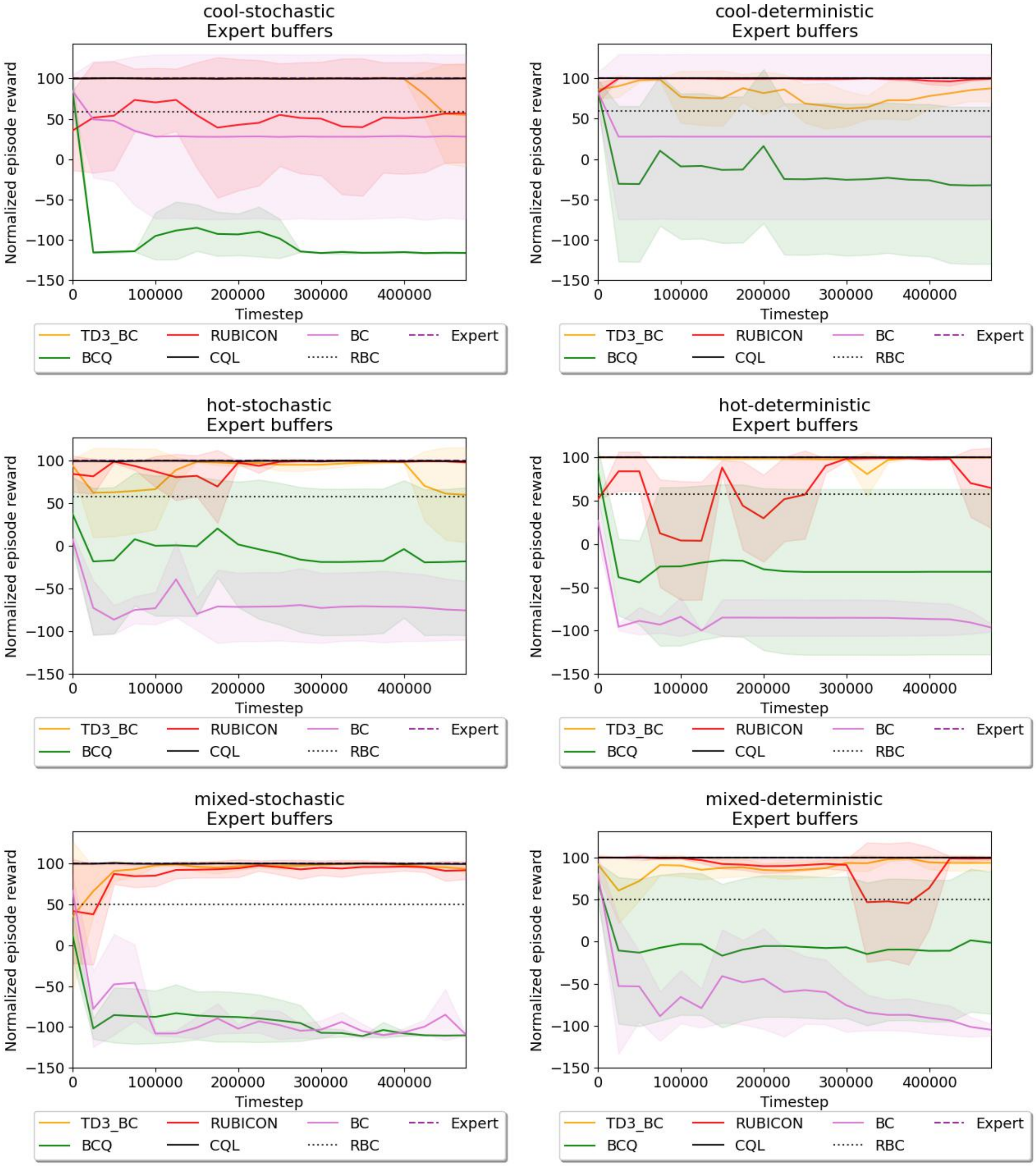


**Figure B.1**: Learning curves of BRL models learn from expert buffers.

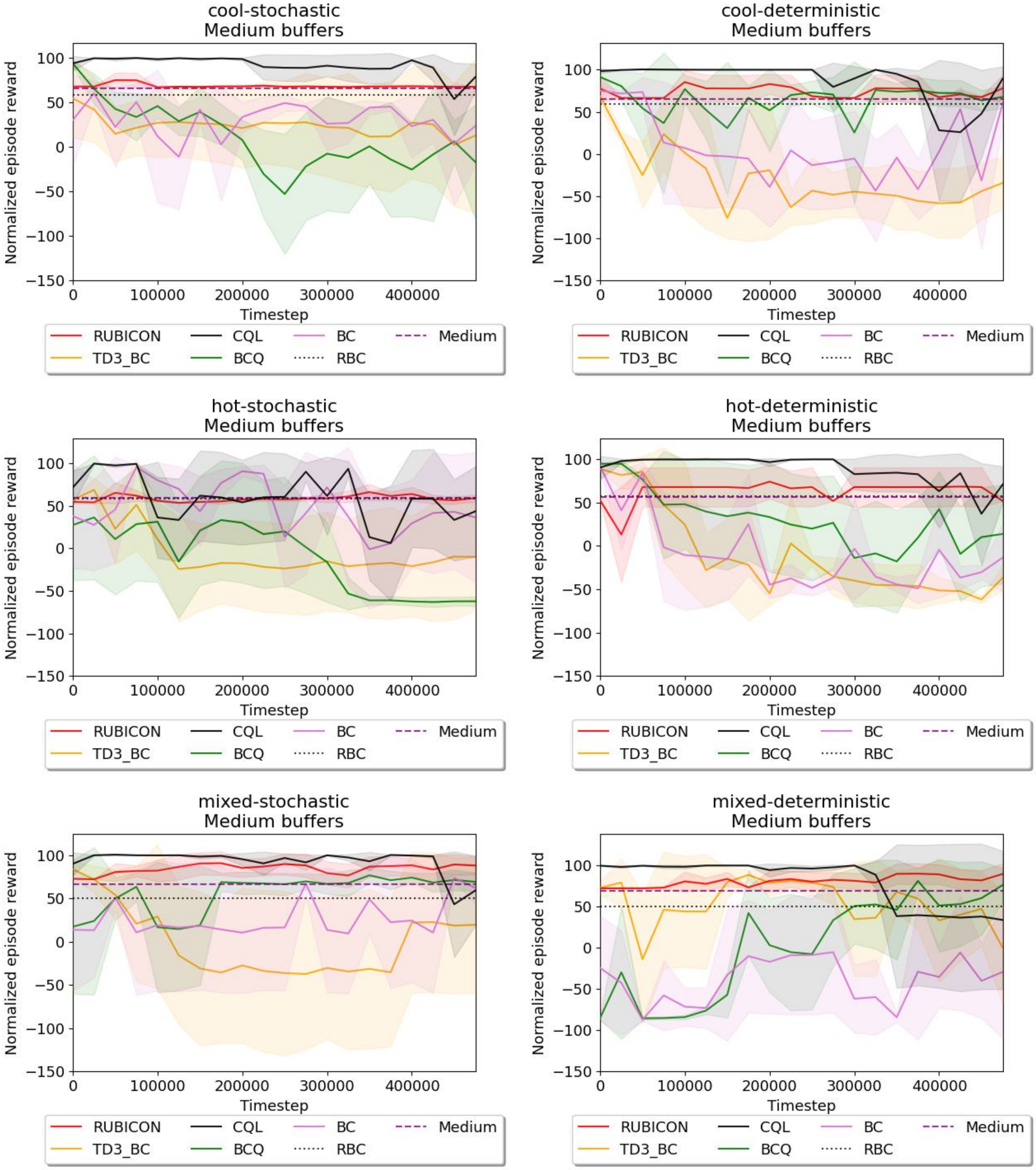


**Figure B.2**: Learning curves of BRL models learn from medium buffers.

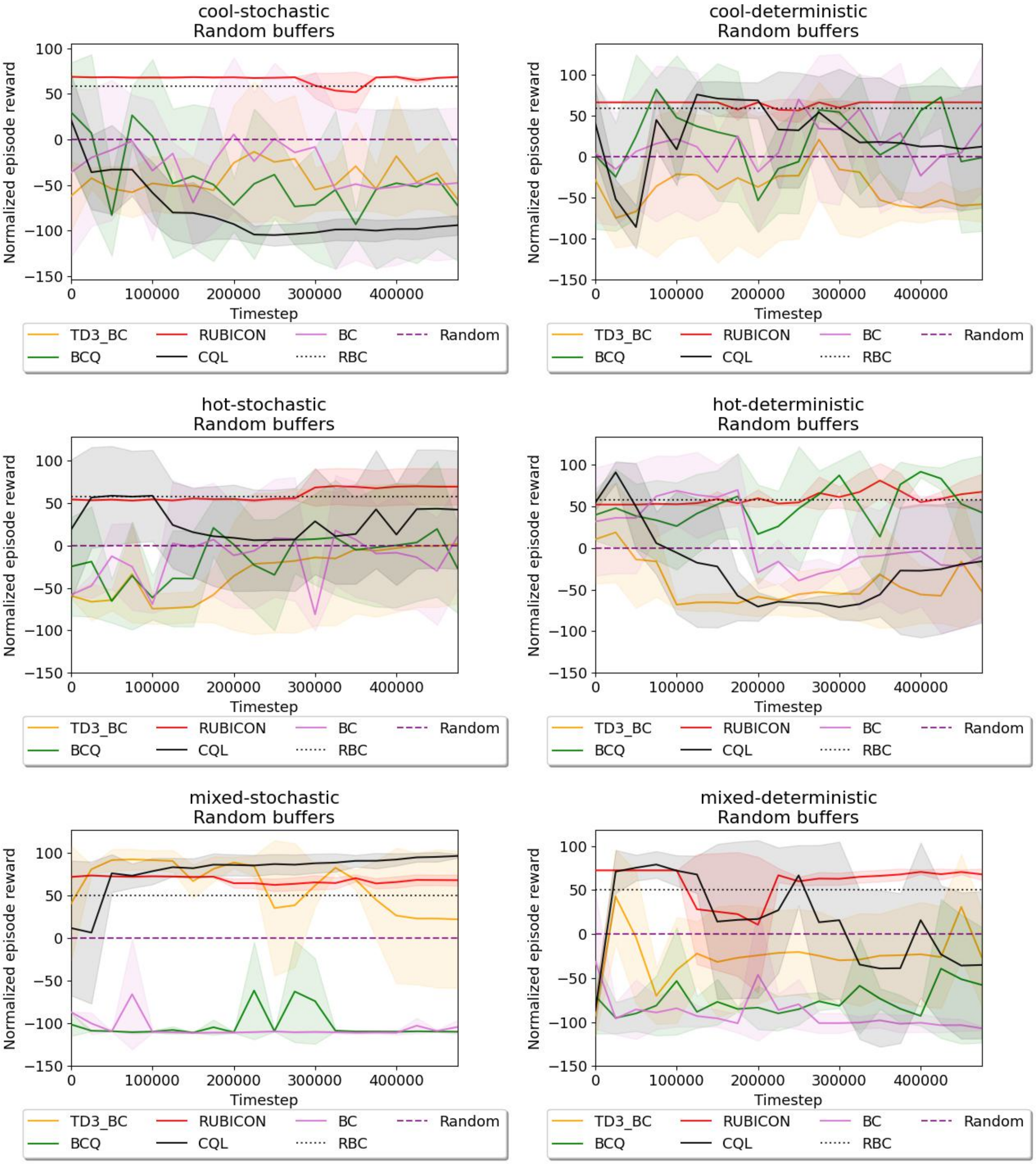
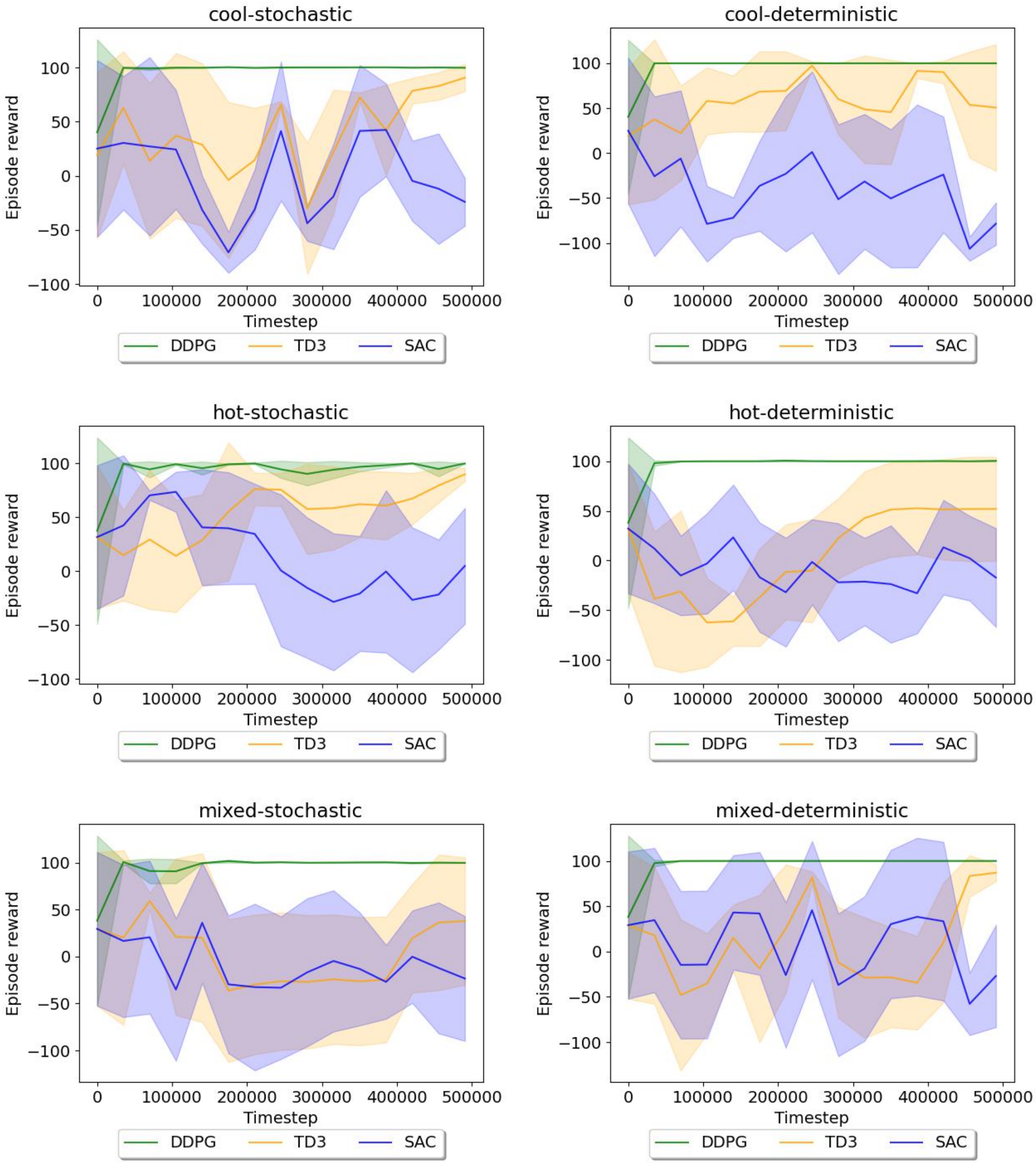


**Figure B.3**: Learning curves of BRL models learn from random buffers


cool-stochastic

cool-deterministic

hot-stochastic

hot-deterministic

mixed-stochastic

mixed-deterministic

**Figure B.4**: Learning curves of behavioral model training, behavioral models are trained with 500K time steps before generating buffers.

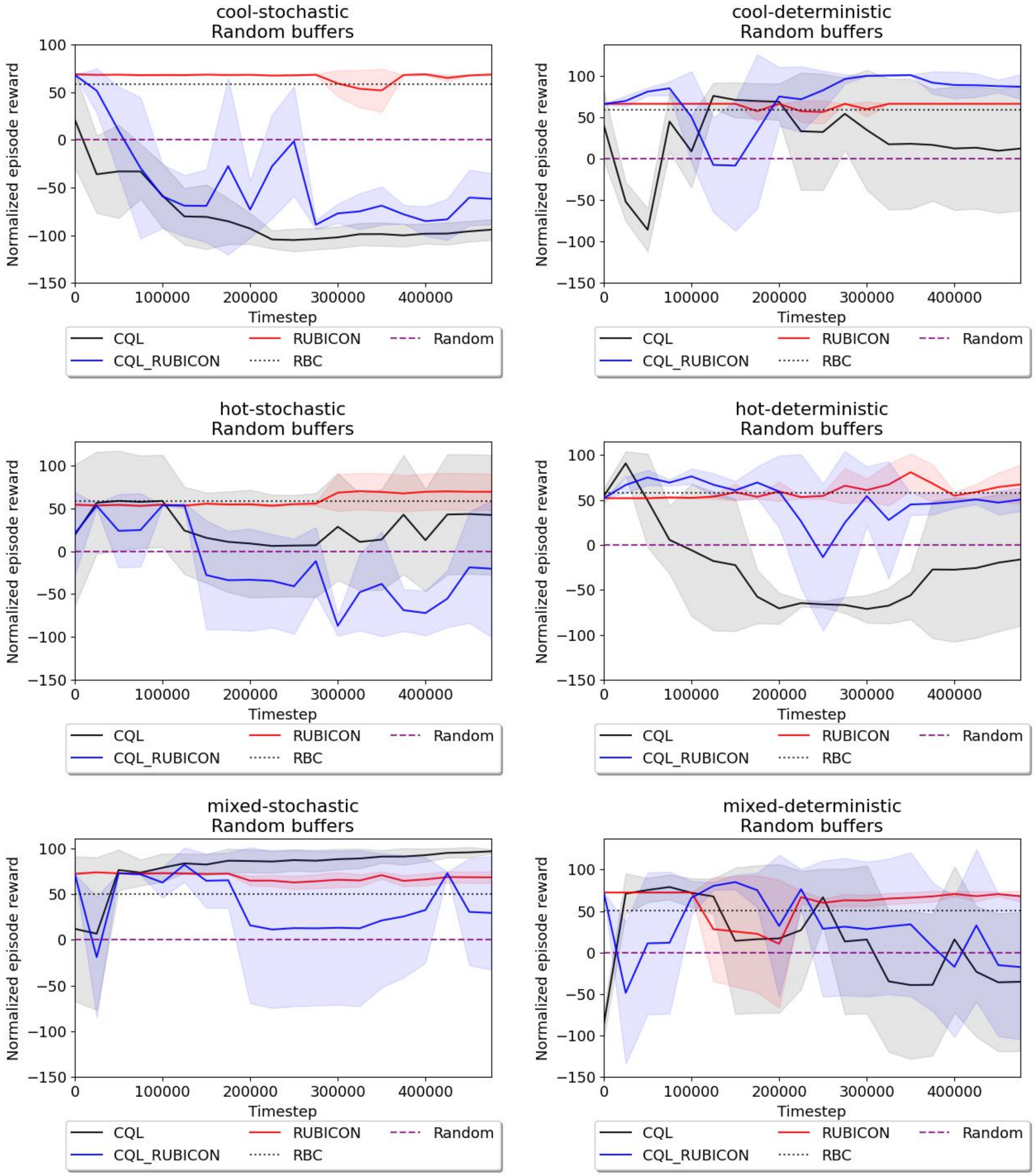


**Figure B.5**: Learning curves of CQL, CQL+RUBICON, and RUBICON learn from random buffers

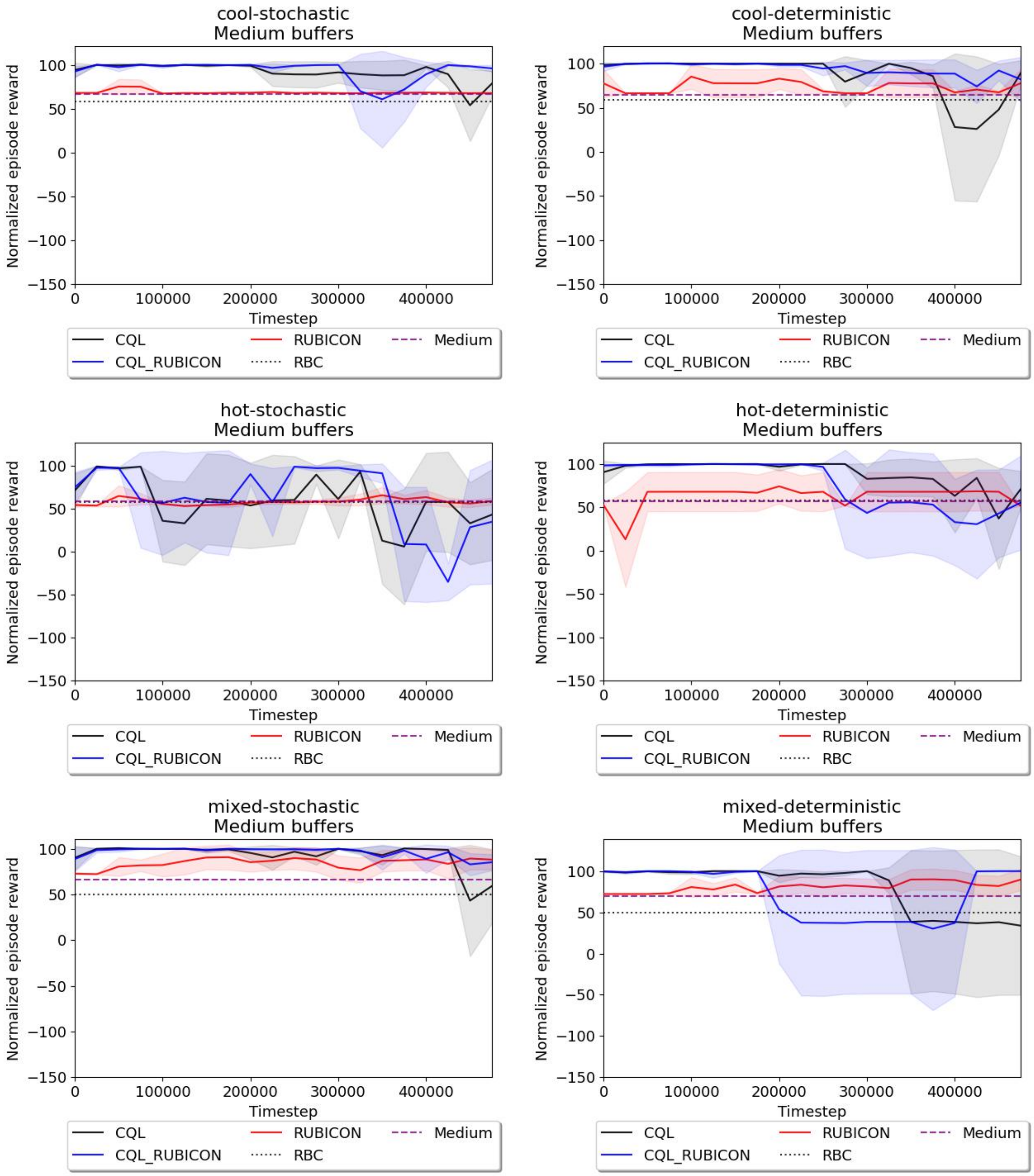


**Figure B.6**: Learning curves of CQL, CQL+RUBICON, and RUBICON learn from medium buffers

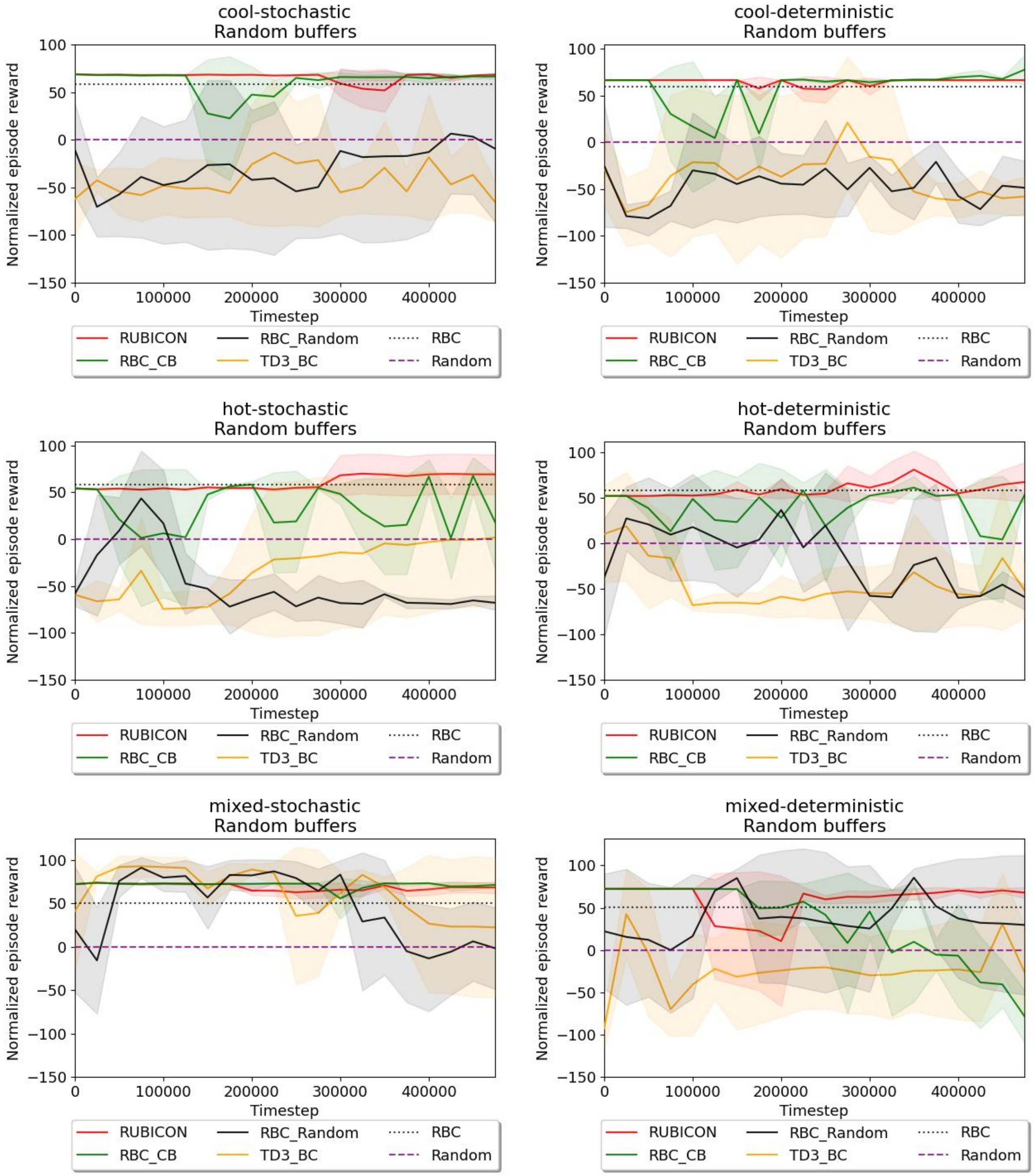


**Figure B.7**: Learning curves of RUBICON learns from worsened RBC compared with TD3+BC and RUBICON

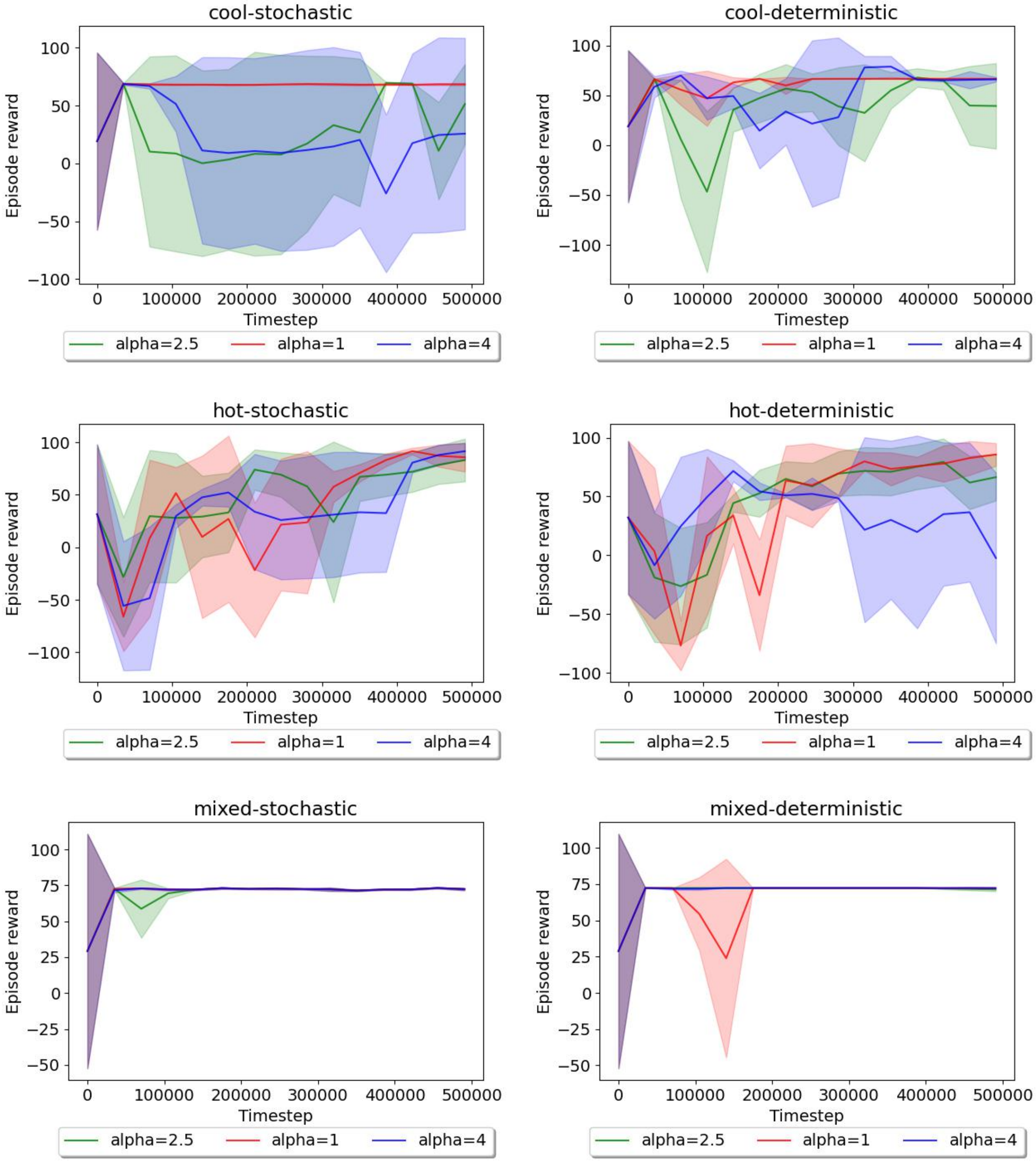


**Figure B.8**: Learning curves of online RUBICON hyperparameter optimization

**Table B.1**: Hyperparameter experiment.

| Environment | $\alpha = 1$ | $\alpha = 2.5$ | $\alpha = 4$ |
|---|---|---|---|
| hot-deterministic | **79.08±12.24** | 70.76±20.28 | 23.83±68.3 |
| mixed-deterministic | **72.34±0.00** | 71.92±0.52 | **72.26±0.05** |
| cool-deterministic | **66.52±0.00** | 53.28±23.81 | **68.16±4.69** |
| hot-stochastic | **83.64±8.25** | 73.92±20.24 | 65.13±28.08 |
| mixed-stochastic | **72.24±0.49** | **72.24±0.49** | **72.24±0.49** |
| cool-stochastic | **68.14±0.65** | 45.46±28.4 | 12.38±77.7 |
| Sum | **441.99±21.64** | 387.58±93.74 | 347.08±130.01 |

**Table B.2**: Transfer experiment

| Environment | Trans. from | RUBICON_Trans. | RUBICON |
|---|---|---|---|
| hot-stochastic | cool-stochastic | **71.42±3.1** | 59.72±5.29 |
| mixed-stochastic | cool-stochastic | 84.93±16.92 | **87.23±12.34** |
| cool-stochastic | hot-stochastic | 54.07±0.85 | **68.07±0.46** |
| mixed-stochastic | hot-stochastic | 81.02±20.09 | **87.23±12.34** |
| cool-stochastic | mixed-stochastic | **72.39±0.68** | 68.07±0.46 |
| hot-stochastic | mixed-stochastic | **75.26±3.3** | 59.72±5.29 |
| Sum | | **439.09±44.94** | 430.04±36.18 |

**Table B.3**: RUBICON learns from buffers generated by RBC compared with RBC buffer performance

| Environment | RUBICON | RBC |
|---|---|---|
| hot-deterministic | **67.92±22.51** | 57.9 |
| mixed-deterministic | **73.68±1.96** | 50.12 |
| cool-deterministic | **72.28±6.53** | 59.15 |
| hot-stochastic | 53.83±0.8 | **57.92** |
| mixed-stochastic | **72.46±0.68** | 50.22 |
| cool-stochastic | 53.35±20.36 | **58.48** |
| Sum | **393.53±52.85** | 333.79 |

**Table B.4**: CQL+RUBICON learns from random buffer compared with CQL and RUBICON

| Environment | RUBICON | CQL_RUBICON | CQL |
|---|---|---|---|
| hot-deterministic | **62.7±14.36** | 48.37±10.12 | -23.19±76.76 |
| mixed-deterministic | **68.83±4.93** | -2±85.18 | -23.46±83.61 |
| cool-deterministic | 66.5±0 | **88.98±13.24** | 12.98±73.04 |
| hot-stochastic | **68.83±21.26** | -47.04±45.48 | 36.64±67.61 |
| mixed-stochastic | 67.03±6.26 | 38.08±49.17 | **94.04±5.87** |
| cool-stochastic | **67.55±1.14** | -73.76±20.51 | -97.35±11.07 |
| Sum | **401.47±47.98** | 52.64±223.72 | -0.32±317.97 |

**Table B.5**: Scores of CQL+RUBICON learns from medium buffer compared with CQL and RUBICON

| Environment | RUBICON | CQL_RUBICON | CQL |
|---|---|---|---|
| hot-deterministic | 64.91±18.02 | 43.03±55.26 | **67.64±32.83** |
| mixed-deterministic | **86.84±12.39** | 73.4±37.59 | 37.36±86.8 |
| cool-deterministic | 72.2±8.07 | **85.24±17.33** | 55.44±49 |
| hot-stochastic | **59.72±5.29** | 9.39±58.72 | 39.92±56.67 |
| mixed-stochastic | 87.23±12.34 | **90.14±9.37** | 80.13±20.78 |
| cool-stochastic | 68.07±0.46 | **91.05±11.39** | 81.56±18.01 |
| Sum | **438.98±56.59** | 392.26±189.69 | 362.07±264.11 |

**Table B.6**: Scores of TD3+BC learns from a mixture of random buffer and RBC buffer compared with RUBICON learns from random buffer

| Environment | TD3+BC_Mixed | RUBICON | TD3+BC |
|---|---|---|---|
| hot-deterministic | 0.02±59.76 | **62.7±14.36** | -45.73±44.8 |
| mixed-deterministic | **70.66±15.45** | 68.83±4.93 | -13.71±57.06 |
| cool-deterministic | 59.01±40.92 | **66.5±0** | -58.4±19.25 |
| hot-stochastic | 57.93±5.6 | **68.83±21.26** | -1.82±73.31 |
| mixed-stochastic | **74.08±8.7** | 67.03±6.26 | 28.01±72.79 |
| cool-stochastic | **71.67±35.04** | 67.55±1.14 | -44.33±36.36 |
| Sum | 333.4±165.5 | **401.47±47.98** | -135.98±303.60 |

**Table B.7**: Comparison between RUBICON, TD3+BC, and worsened RBCs

| Environment | RUBICON | TD3+BC | RBC_CB | RBC_Random |
|---|---|---|---|---|
| hot-deterministic | **62.7±14.36** | -45.73±44.8 | 34.06±27.13 | -47.72±25.11 |
| mixed-deterministic | **68.83±4.93** | -13.71±57.06 | -33.89±38.47 | 36.39±72.08 |
| cool-deterministic | 66.5±0 | -58.4±19.25 | **70.64±5.85** | -48.84±25.93 |
| hot-stochastic | **68.83±21.26** | -1.82±73.31 | 33.81±36.94 | -67.7±5.66 |
| mixed-stochastic | 67.03±6.26 | 28.01±72.79 | **71.22±2.64** | -4.07±52.46 |
| cool-stochastic | **67.55±1.14** | -44.33±36.36 | 65.84±3.06 | -5.9±74.12 |
| Sum | **401.47±47.98** | -135.98±303.60 | 241.69±114.12 | -137.85±255.38 |

## B.3 Model parameters

We list the hyperparameters used in this paper for reproducibility. Unless mentioned otherwise, we keep the original hyperparameters setups as the implementations listed in Appendix B.1 since DRL methods are sensitive to hyperparameter tuning [43] (see Table B.9, B.10, B.11, and B.12).

**Table B.8**: Non-selective experiment

| Environment | Buffer | RUBICON | RUBICON w/o DW |
|---|---|---|---|
| hot-deterministic | Expert | **86.13±17.83** | -19.8±63.89 |
| hot-deterministic | Medium | **64.91±18.02** | 47.26±12.89 |
| hot-deterministic | Random | **62.7±14.36** | -19.8±63.89 |
| mixed-deterministic | Expert | **81±25.94** | -75.6±29.46 |
| mixed-deterministic | Medium | **86.84±12.39** | 42.99±48.04 |
| mixed-deterministic | Random | **68.83±4.93** | -75.6±29.46 |
| cool-deterministic | Expert | **98±2.78** | 41.3±20.53 |
| cool-deterministic | Medium | **72.2±8.07** | 36.54±67.84 |
| cool-deterministic | Random | **66.5±0** | 41.3±20.53 |
| hot-stochastic | Expert | **99.01±0.56** | 57.68±22.3 |
| hot-stochastic | Medium | **59.72±5.29** | 29.26±45.5 |
| hot-stochastic | Random | **68.83±21.26** | 57.68±22.3 |
| mixed-stochastic | Expert | **94.16±8.12** | 40.57±44.91 |
| mixed-stochastic | Medium | **87.23±12.34** | 55.6±33.53 |
| mixed-stochastic | Random | **67.03±6.26** | 40.57±44.91 |
| cool-stochastic | Expert | **53.58±65.53** | -68.84±27.46 |
| cool-stochastic | Medium | **68.07±0.46** | 8.01±61.1 |
| cool-stochastic | Random | **67.55±1.14** | -68.84±27.46 |
| Sum | | **1352.37±225.38** | 170.3±686.1 |

**Table B.9**: TD3, TD3+BC, and RUBICON hyperparameters

| | Hyperparameter | Value |
|---|---|---|
| Algorithm hyperparameters | Optimizer | Adam [50] |
| | Critic learning rate | $3e^{-4}$ |
| | Actor learning rate | $3e^{-4}$ |
| | Mini-batch size | 256 |
| | Discount factor | 0.99 |
| | Target update rate | $5e^{-3}$ |
| | Policy noise | 0.2 |
| | Policy noise clipping | (-0.5, 0.5) |
| | Policy update frequency | 2 |
| | TD3+BC $\alpha$ | 2.5 |
| | RUBICON online $\alpha$ | 1 |
| | RUBICON offline $\alpha$ | 2.5 |
| | RUBICON online $\xi$ | 0 if $\bar{Q}(s, \pi_b(s)) \geq \bar{Q}(s, \pi_{rbc}(s))$ else 1 |
| | RUBICON offline $\xi$ | 1 |
| Network architecture | Critic hidden dimension | 256 |
| | Critic hidden layers | 2 |
| | Critic activation function | ReLU |
| | Actor hidden dimension | 256 |
| | Actor hidden layers | 2 |
| | Actor activation function | ReLU |

**Table B.10**: SAC/CQL hyperparameters

| | Hyperparameter | Value |
|---|---|---|
| Algorithm hyperparameters | Optimizer | Adam |
| | Critic learning rate | $1e^{-3}$ |
| | Actor learning rate | $3e^{-4}/1e^{-4}$ |
| | Mini-batch size | 256 |
| | Discount factor | 0.99 |
| | Target update rate | $5e^{-3}$ |
| | Policy noise | 0.2 |
| | Policy noise clipping | (-0.5, 0.5) |
| | Policy update frequency | 2 |
| | SAC entropy auto-tuning | True |
| | CQL $\alpha$ threshold | 10 |
| | CQL conservative weight | 5.0 |
| | CQL number of sampled actions | 10 |
| Network architecture | Critic hidden dimension | 256 |
| | Critic hidden layers | 3 |
| | Critic activation function | ReLU |
| | Actor hidden dimension | 256 |
| | Actor hidden layers | 3 |
| | Actor activation function | ReLU |

**Table B.11**: DDPG hyperparameters

| | Hyperparameter | Value |
|---|---|---|
| Algorithm hyperparameters | Optimizer | Adam |
| | Critic learning rate | $1e^{-3}$ |
| | Actor learning rate | $1e^{-4}$ |
| | Mini-batch size | 64 |
| | Discount factor | 0.99 |
| | Target update rate | $1e^{-3}$ |
| | Policy noise | $\mathcal{N}(0, 0.1)$ |
| | Policy noise clipping | (-0.5, 0.5) |
| | Policy update frequency | 1 |
| Network architecture | Critic hidden dimension | 400/300 |
| | Critic hidden layers | 2 |
| | Critic activation function | ReLU |
| | Actor hidden dimension | 400/300 |
| | Actor hidden layers | 2 |
| | Actor activation function | ReLU |

**Table B.12**: BCQ/BC hyperparameters

| | Hyperparameter | Value |
|---|---|---|
| Algorithm hyperparameters | Optimizer | Adam |
| | Critic learning rate | $1e^{-3}$ |
| | Actor learning rate | $1e^{-4}$ |
| | Mini-batch size | 100 |
| | Discount factor | 0.99 |
| | Target update rate | $5e^{-3}$ |
| | Minimum weighting | 0.75 |
| | Max perturbation | 0.05 |
| Network architecture | Critic hidden dimension | 400/300 |
| | Critic hidden layers | 2 |
| | Critic activation function | ReLU |
| | Actor hidden dimension | 400/300 |
| | Actor hidden layers | 2 |
| | Actor activation function | ReLU |
| | VAE hidden dimension | 750 |
| | VAE latent vector clipping | (-0.5, 0.5) |

# Appendix C

# Adaptive Policy Regularization for Offline-to-Online Reinforcement Learning in HVAC Control

## C.1 RL Setup of Data Center Environment

- **State**: Year, month, day, hour, site outdoor air drybulb temperature(Environment), site outdoor air relative humidity(Environment), site wind speed(Environment), site wind direction(Environment), site diffuse solar radiation rate per area(Environment), site direct solar tadiation tate per area(Environment), zone thermostat heating setpoint temperature(West Zone), zone thermostat cooling setpoint temperature(West Zone), zone air temperature(West Zone), zone thermal comfort mean radiant temperature(West Zone PEOPLE), zone air relative humidity(West Zone), zone thermal comfort clothing value(West Zone PEOPLE), zone thermal comfort Fanger model PPD(West Zone PEOPLE), zone people occupant count(West Zone), people air temperature(West Zone PEOPLE), zone thermostat heating setpoint temperature(East Zone), zone thermostat cooling setpoint temperature(East Zone), zone air temperature(East Zone), zone thermal comfort mean radiant temperature(East Zone PEOPLE), zone air relative humidity(East Zone), zone thermal comfort clothing value(East Zone PEOPLE), zone thermal comfort Fanger model PPD(East Zone PEOPLE), zone people occupant count(East Zone), people air temperature(East Zone PEOPLE), and facility total HVAC electricity demand rate(Whole Building)

- **Action**: Heating setpoint and cooling setpoint of West and East zones in continuous settings for the interior zones.
- **Reward**: We follow the default linear reward setting, which considers the energy consumption and the absolute difference to temperature comfort.

# Bibliography

[1] Tameem Adel, Alexander Rosenberg, and Been Kim. Learning to explain: An information-theoretic perspective on model interpretation. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 10027–10036, 2019.

[2] Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron Courville, and Marc G Bellemare. Deep reinforcement learning at the edge of the statistical precipice. *arXiv preprint arXiv:2108.13264*, 2021.

[3] Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron C Courville, and Marc Bellemare. Deep reinforcement learning at the edge of the statistical precipice. *Advances in neural information processing systems*, 34:29304–29320, 2021.

[4] Mohammed Alshiekh, Roderick Bloem, Rüdiger Ehlers, Bettina Könighofer, Scott Niekum, and Ufuk Topcu. Safe reinforcement learning via shielding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[5] Anil Aswani, Neal Master, Jay Taneja, David Culler, and Claire Tomlin. Reducing transient and steady state electricity consumption in hvac using learning-based model-predictive control. *Proceedings of the IEEE*, 100(1):240–253, 2011.

[6] Bharathan Balaji, Hidetoshi Teraoka, Rajesh Gupta, and Yuvraj Agarwal. Zonepac: Zonal power estimation and control via hvac metering and occupant feedback. In *BuildSys*, pages 1–8, 2013.

[7] Philip J Ball, Laura Smith, Ilya Kostrikov, and Sergey Levine. Efficient online reinforcement learning with offline data. *arXiv preprint arXiv:2302.02948*, 2023.

[8] Philip J Ball, Laura Smith, Ilya Kostrikov, and Sergey Levine. Efficient online reinforcement learning with offline data. *arXiv preprint arXiv:2302.02948*, 2023.

[9] Farinaz Behrooz, Norman Mariun, Mohammad Hamiruce Marhaban, Mohd Amran Mohd Radzi, and Abdul Rahman Ramli. Review of control techniques for hvac systems—nonlinearity approaches based on fuzzy cognitive maps. *Energies*, 11(3):495, 2018.

[10] Alex Beltran and Alberto E Cerpa. Optimal hvac building control with occupancy prediction. In *BuildSys*, pages 168–171, 2014.

[11] Hamid R Berenji. A reinforcement learning—based architecture for fuzzy logic control. *International Journal of Approximate Reasoning*, 6(2):267–292, 1992.

[12] Donald J Berndt and James Clifford. Using dynamic time warping to find patterns in time series. In *KDD workshop*, volume 10, pages 359–370. Seattle, WA, USA:, 1994.

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

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

[15] Francesco Calvino, Maria La Gennusa, Gianfranco Rizzo, and Gianluca Scaccianoce. The control of indoor thermal comfort conditions: introducing a fuzzy adaptive controller. *Energy and buildings*, 36(2):97–102, 2004.

[16] CDC. Guidelines for environmental infection control in health-care facilities. `https://www.cdc.gov/infectioncontrol/guidelines/environmental/background/air.html`, 2003.

[17] Bingqing Chen, Zicheng Cai, and Mario Bergés. Gnu-rl: A precocial reinforcement learning solution for building hvac control using a differentiable mpc policy. In *BuildSys*, pages 316–325, 2019.

[18] Drury B Crawley, Linda K Lawrie, Frederick C Winkelmann, Walter F Buhl, Y Joe Huang, Curtis O Pedersen, Richard K Strand, Richard J Liesen, Daniel E Fisher, Michael J Witte, et al. Energyplus: creating a new-generation building energy simulation program. *Energy and buildings*, 33(4):319–331, 2001.

[19] Hui Dai and Bin Zhao. Association of the infection probability of covid-19 with ventilation rates in confined spaces. In *Building simulation*, volume 13, 2020.

[20] Megan Dawe, Paul Raftery, Jonathan Woolley, Stefano Schiavon, and Fred Bauman. Comparison of mean radiant and air temperatures in mechanically-conditioned commercial buildings from over 200,000 field and laboratory measurements. *Energy and Buildings*, 206:109582, 2020.

[21] Richard J De Dear. A global database of thermal comfort field experiments. *ASHRAE transactions*, 104:1141, 1998.

[22] Adithya M Devraj, Ana Bušić, and Sean Meyn. Zap q-learning-a user's guide. In *2019 Fifth Indian Control Conference (ICC)*, pages 10–15. IEEE, 2019.

[23] Xianzhong Ding, Wan Du, and Alberto Cerpa. Octopus: Deep reinforcement learning for holistic smart building control. In *BuildSys*, pages 326–335, 2019.

[24] Xianzhong Ding, Wan Du, and Alberto E Cerpa. Mb2c: Model-based deep reinforcement learning for multi-zone building control. In *BuildSys*, pages 50–59, 2020.

[25] Anastasios I Dounis, M Bruant, M Santamouris, G Guarracino, and P Michel. Comparison of conventional and fuzzy control of indoor air quality in buildings. *Journal of Intelligent & Fuzzy Systems*, 4(2):131–140, 1996.

[26] Gabriel Dulac-Arnold, Daniel Mankowitz, and Todd Hester. Challenges of real-world reinforcement learning. *arXiv preprint arXiv:1904.12901*, 2019.

[27] Povl O Fanger et al. Thermal comfort. analysis and applications in environmental engineering. *Thermal comfort. Analysis and applications in environmental engineering.*, 1970.

[28] William Fedus, Prajit Ramachandran, Rishabh Agarwal, Yoshua Bengio, Hugo Larochelle, Mark Rowland, and Will Dabney. Revisiting fundamentals of experience replay. In *International Conference on Machine Learning*, pages 3061–3071. PMLR, 2020.

[29] Arduin Findeis, Fiodar Kazhamiaka, Scott Jeen, and Srinivasan Keshav. Beobench: a toolkit for unified access to building simulations for reinforcement learning. In *Proceedings of the Thirteenth ACM International Conference on Future Energy Systems*, pages 374–382, 2022.

[30] Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

[31] Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

[32] Xiaohan Fu, Jason Koh, Francesco Fraternali, Dezhi Hong, and Rajesh Gupta. Zonal air handling in commercial buildings. In *BuildSys*, pages 302–303, 2020.

[33] Scott Fujimoto and Shixiang Shane Gu. A minimalist approach to offline reinforcement learning. *Advances in neural information processing systems*, 34:20132–20145, 2021.

[34] Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *ICML*, pages 1587–1596. PMLR, 2018.

[35] Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pages 1587–1596. PMLR, 2018.

[36] Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In *ICML*, pages 2052–2062. PMLR, 2019.

[37] Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In *ICML*, pages 2052–2062. PMLR, 2019.

[38] Guanyu Gao, Jie Li, and Yonggang Wen. Deepcomfort: Energy-efficient thermal comfort control in buildings via reinforcement learning. *IEEE Internet of Things Journal*, 7(9):8472–8484, 2020.

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

[40] Mengjie Han, Ross May, Xingxing Zhang, Xinru Wang, Song Pan, Da Yan, Yuan Jin, and Liguo Xu. A review of reinforcement learning methodologies for controlling occupant comfort in buildings. *Sustainable Cities and Society*, 51:101748, 2019.

[41] Frederick Hayes-Roth. Rule-based systems. *Communications of the ACM*, 28(9):921–932, 1985.

[42] Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. In *AAAI*, number 1, 2018.

[43] Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[44] Todd Hester, Matej Vecerik, Olivier Pietquin, Marc Lanctot, Tom Schaul, Bilal Piot, Dan Horgan, John Quan, Andrew Sendonaris, Ian Osband, et al. Deep q-learning from demonstrations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[45] Shengyi Huang, Rousslan Fernand Julien Dossa, Chang Ye, and Jeff Braga. Cleanrl: High-quality single-file implementations of deep reinforcement learning algorithms. *arXiv preprint arXiv:2111.08819*, 2021.

[46] Lu Jiang, Tong Xiao, and Thomas Huang. Learning to explain: A framework for machine learning explanations. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 9810–9820, 2018.

[47] Javier Jiménez-Raboso, Alejandro Campoy-Nieves, Antonio Manjavacas-Lucas, Juan Gómez-Romero, and Miguel Molina-Solana. Sinergym: a building simulation and control framework for training reinforcement learning agents. In *Proceedings of the 8th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation*, pages 319–323, 2021.

[48] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. *NIPS*, 30:3146–3154, 2017.

[49] Beomjoon Kim, Amir-massoud Farahmand, Joelle Pineau, and Doina Precup. Learning from limited demonstrations. *Advances in Neural Information Processing Systems*, 26, 2013.

[50] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[51] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[52] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[53] SA Klein. University of wisconsin-madison solar energy laboratory. *TRNSYS: A transient simulation program. Eng. Experiment Station*, 1976.

[54] Ilya Kostrikov, Rob Fergus, Jonathan Tompson, and Ofir Nachum. Offline reinforcement learning with fisher divergence critic regularization. In *International Conference on Machine Learning*, pages 5774–5783. PMLR, 2021.

[55] Andrew Krioukov, Gabe Fierro, Nikita Kitaev, and David Culler. Building application stack (bas). In *BuildSys*, pages 72–79, 2012.

[56] Solomon Kullback and Richard A Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951.

[57] Aviral Kumar, Justin Fu, Matthew Soh, George Tucker, and Sergey Levine. Stabilizing off-policy q-learning via bootstrapping error reduction. *Advances in Neural Information Processing Systems*, 32, 2019.

[58] Aviral Kumar, Justin Fu, George Tucker, and Sergey Levine. Stabilizing off-policy q-learning via bootstrapping error reduction. *arXiv preprint arXiv:1906.00949*, 2019.

[59] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 33:1179–1191, 2020.

[60] National Renewable Energy Laboratory. Tmy3 datasets. `https://www.nrel.gov/docs/fy08osti/43156.pdf`, 2008.

[61] Daeil Lee, Awwal Mohammed Arigi, and Jonghyun Kim. Algorithm for autonomous power-increase operation using deep reinforcement learning and a rule-based system. *IEEE Access*, 8:196727–196746, 2020.

[62] Seunghyun Lee, Younggyo Seo, Kimin Lee, Pieter Abbeel, and Jinwoo Shin. Offline-to-online reinforcement learning via balanced replay and pessimistic q-ensemble. In *Conference on Robot Learning*, pages 1702–1712. PMLR, 2022.

[63] Geoff J Levermore. Building energy management systems. 1992.

[64] Yuguo Li, Hua Qian, Jian Hang, Xuguang Chen, Ling Hong, Peng Liang, Jiansen Li, Shenglan Xiao, Jianjian Wei, Li Liu, et al. Evidence for probable aerosol transmission of sars-cov-2 in a poorly ventilated restaurant. *MedRxiv*, 2020.

[65] Amarildo Likmeta, Alberto Maria Metelli, Andrea Tirinzoni, Riccardo Giol, Marcello Restelli, and Danilo Romano. Combining reinforcement learning with rule-based controllers for transparent and general decision-making in autonomous driving. *Robotics and Autonomous Systems*, 131:103568, 2020.

[66] 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. *arXiv preprint arXiv:1509.02971*, 2015.

[67] Long-Ji Lin. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine learning*, 8(3):293–321, 1992.

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

[69] Hsin-Yu Liu, Xiaohan Fu, Bharathan Balaji, Rajesh Gupta, and Dezhi Hong. B2rl: an open-source dataset for building batch reinforcement learning. In *Proceedings of the 9th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation*, pages 462–465, 2022.

[70] Yao Liu, Adith Swaminathan, Alekh Agarwal, and Emma Brunskill. Provably good batch reinforcement learning without great exploration. *arXiv preprint arXiv:2007.08202*, 2020.

[71] Siliang Lu, Weilong Wang, Chaochao Lin, and Erica Cochran Hameen. Data-driven simulation of a thermal comfort-based temperature set-point control with ashrae rp884. *Building and Environment*, 156:137–146, 2019.

[72] Yicheng Luo, Jackie Kay, Edward Grefenstette, and Marc Peter Deisenroth. Finetuning from offline reinforcement learning: Challenges, trade-offs and practical solutions. *arXiv preprint arXiv:2303.17396*, 2023.

[73] Mehdi Maasoumy, Alessandro Pinto, and Alberto Sangiovanni-Vincentelli. Model-based hierarchical optimal control design for hvac systems. In *Dynamic Systems and Control Conference*, volume 54754, pages 271–278, 2011.

[74] Mehdi Maasoumy, M Razmara, M Shahbakhti, and A Sangiovanni Vincentelli. Handling model uncertainty in model predictive control for energy efficient buildings. *Energy and Buildings*, 77:377–392, 2014.

[75] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *ICML*, pages 1928–1937. PMLR, 2016.

[76] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[77] Srinarayana Nagarathinam, Vishnu Menon, Arunchandar Vasan, and Anand Sivasubramaniam. Marco-multi-agent reinforcement learning based control of building hvac systems. In *e-Energy*, pages 57–67, 2020.

[78] Ashvin Nair, Abhishek Gupta, Murtaza Dalal, and Sergey Levine. Awac: Accelerating online reinforcement learning with offline datasets. *arXiv preprint arXiv:2006.09359*, 2020.

[79] Department of Energy. Prototype building models, 2023.

[80] Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. Deep exploration via bootstrapped dqn. *Advances in neural information processing systems*, 29, 2016.

[81] June Young Park and Zoltan Nagy. Hvaclearn: A reinforcement learning based occupant-centric control for thermostat set-points. In *e-Energy*, pages 434–437, 2020.

[82] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[83] Luis Perez-Lombard, Jose Ortiz, and Christine Pout. A review on buildings energy consumption information. *Energy and Buildings*, 40(3):394–398, 2008.

[84] Bilal Piot, Matthieu Geist, and Olivier Pietquin. Boosted bellman residual minimization handling expert demonstrations. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2014, Nancy, France, September 15-19, 2014. Proceedings, Part II 14*, pages 549–564. Springer, 2014.

[85] Samuel Prívara, Zdeněk Váňa, Dimitrios Gyalistras, Jiří Cigler, Carina Sagerschnig, Manfred Morari, and Lukáš Ferkl. Modeling and identification of a large multi-zone office building. In *2011 IEEE International Conference on Control Applications (CCA)*, pages 55–60. IEEE, 2011.

[86] Majdi I Radaideh and Koroush Shirvan. Rule-based reinforcement learning methodology to inform evolutionary algorithms for constrained optimization of engineering applications. *Knowledge-Based Systems*, 217:106836, 2021.

[87] Naren Srivaths Raman, Adithya M Devraj, Prabir Barooah, and Sean P Meyn. Reinforcement learning for control of building hvac systems. In *2020 American Control Conference (ACC)*, pages 2326–2332. IEEE, 2020.

[88] REHVA. Rehva covid19 guidance v4.1. `https://www.rehva.eu/fileadmin/user_upload/REHVA_COVID-19_guidance_document_V4.1_15042021.pdf`, 2021.

[89] Martin Riedmiller. Neural fitted q iteration–first experiences with a data efficient neural reinforcement learning method. In *ECML*, pages 317–328. Springer, 2005.

[90] Brian C Ross. Mutual information between discrete and continuous data sets. *PloS one*, 9(2):e87357, 2014.

[91] Frederik Ruelens, Bert J Claessens, Stijn Vandael, Bart De Schutter, Robert Babuška, and Ronnie Belmans. Residential demand response of thermostatically controlled loads using batch reinforcement learning. *IEEE Transactions on Smart Grid*, 8(5):2149–2159, 2016.

[92] Frederik Ruelens, Bert J Claessens, Stijn Vandael, Sandro Iacovella, Pieter Vingerhoets, and Ronnie Belmans. Demand response of a heterogeneous cluster of electric water heaters using batch reinforcement learning. In *2014 Power Systems Computation Conference*, pages 1–7. IEEE, 2014.

[93] Frederik Ruelens, Sandro Iacovella, Bert J Claessens, and Ronnie Belmans. Learning agent for a heat-pump thermostat with a set-back strategy using model-free reinforcement learning. *Energies*, 8(8):8300–8318, 2015.

[94] Jyri Salpakari and Peter Lund. Optimal and rule-based control strategies for energy flexibility in buildings with pv. *Applied Energy*, 161:425–436, 2016.

[95] Caude Sammut. *Behavioral Cloning*, pages 93–97. Springer US, 2010.

[96] Paul Scharnhorst, Baptiste Schubnel, Carlos Fernández Bandera, Jaume Salom, Paolo Taddeo, Max Boegli, Tomasz Gorecki, Yves Stauffer, Antonis Peppas, and Chrysa Politi. Energym: A building model library for controller benchmarking. *Applied Sciences*, 11(8):3518, 2021.

[97] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*, 2015.

[98] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. In *International Conference on Learning Representations (ICLR)*, 2016.

[99] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897. PMLR, 2015.

[100] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[101] Takuma Seno and Michita Imai. d3rlpy: An offline deep reinforcement learning library. *arXiv preprint arXiv:2111.03788*, 2021.

[102] AB Shepherd and WJ Batty. Fuzzy control strategies to provide cost and energy efficient high quality indoor environments in buildings with high occupant densities. *Building Services Engineering Research and Technology*, 24(1), 2003.

[103] Ravid Shwartz-Ziv and Amitai Armon. Tabular data: Deep learning is not all you need. *arXiv preprint arXiv:2106.03253*, 2021.

[104] David Silver. Lecture 3: Planning by dynamic programming. *UCL Course on RL*, 2015.

[105] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. In *International conference on machine learning*, pages 387–395. PMLR, 2014.

[106] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359, 2017.

[107] Przemysław Spurek, Damian Szymański, and Tomasz Tajmajer. Towards interpretable reinforcement learning using attention augmented agents. In *Proceedings of the 2019 International Conference on Robotics and Automation (ICRA)*, pages 3239–3245. IEEE, 2019.

[108] Yanan Sui, Alkis Gotovos, Joel Burdick, and Andreas Krause. Safe exploration for optimization with gaussian processes. In *International conference on machine learning*, pages 997–1005. PMLR, 2015.

[109] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[110] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[111] Muthusamy V Swami and Subrato Chandra. Procedures for calculating natural ventilation airflow rates in buildings. *ASHRAE final report FSEC-CR-163-86, ASHRAE research project*, page 130, 1987.

[112] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pages 5026–5033. IEEE, 2012.

[113] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012.

[114] Matteo Turchetta, Felix Berkenkamp, and Andreas Krause. Safe exploration in finite markov decision processes with gaussian processes. *Advances in Neural Information Processing Systems*, 29, 2016.

[115] IEA UN. Global status report for buildings and construction (2019). *Avail-able at https://www. gbpn. org/china/newsroom/2019-global-status-report-buildings-and-constr uction. Access date*, 15, 2020.

[116] William Valladares, Marco Galindo, Jorge Gutiérrez, Wu-Chieh Wu, Kuo-Kai Liao, Jen-Chung Liao, Kuang-Chin Lu, and Chi-Chuan Wang. Energy optimization associated with thermal comfort and indoor air control via a deep reinforcement learning algorithm. *Building and Environment*, 155:105–117, 2019.

[117] Stefan Van Der Walt, S Chris Colbert, and Gael Varoquaux. The numpy array: a structure for efficient numerical computation. *Computing in science & engineering*, 13(2):22–30, 2011.

[118] José Vázquez-Canteli, Jérôme Kämpf, and Zoltán Nagy. Balancing comfort and energy consumption of a heat pump using batch reinforcement learning with fitted q-iteration. *Energy Procedia*, 122:415–420, 2017.

[119] José Vázquez-Canteli, Stepan Ulyanin, Jérôme Kämpf, and Zoltán Nagy. Adaptive multi-agent control of hvac systems for residential demand response using batch reinforcement learning. 2018.

[120] José R Vázquez-Canteli, Sourav Dey, Gregor Henze, and Zoltán Nagy. Citylearn: Standardizing research in multi-agent reinforcement learning for demand response and urban energy management. *arXiv preprint arXiv:2012.10504*, 2020.

[121] Nino Vieillard, Tadashi Kozuno, Bruno Scherrer, Olivier Pietquin, Rémi Munos, and Matthieu Geist. Leverage the average: an analysis of kl regularization in reinforcement learning. In *NeurIPS*, 2020.

[122] Nino Vieillard, Olivier Pietquin, and Matthieu Geist. Munchausen reinforcement learning. *arXiv preprint arXiv:2007.14430*, 2020.

[123] Nino Vieillard, Olivier Pietquin, and Matthieu Geist. Munchausen reinforcement learning. *Advances in Neural Information Processing Systems*, 33:4235–4246, 2020.

[124] Nino Vieillard, Olivier Pietquin, and Matthieu Geist. Munchausen reinforcement learning. *arXiv preprint arXiv:2007.14430*, 2020.

[125] Junjie Wang, Qichao Zhang, Dongbin Zhao, and Yaran Chen. Lane change decision-making through deep reinforcement learning with rule-based constraints. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–6. IEEE, 2019.

[126] Zhe Wang and Tianzhen Hong. Reinforcement learning for building controls: The opportunities and challenges. *Applied Energy*, 269:115036, 2020.

[127] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8:279–292, 1992.

[128] Tianshu Wei, Yanzhi Wang, and Qi Zhu. Deep reinforcement learning for building hvac control. In *Proceedings of the 54th annual design automation conference 2017*, pages 1–6, 2017.

[129] Tianshu Wei, Yanzhi Wang, and Qi Zhu. Deep reinforcement learning for building hvac control. In *Proceedings of the 54th annual design automation conference 2017*, pages 1–6, 2017.

[130] Thomas Weng, Anthony Nwokafor, and Yuvraj Agarwal. Buildingdepot 2.0: An integrated management system for building analysis and control. In *Proceedings of the 5th ACM Workshop on Embedded Systems For Energy-Efficient Buildings*, pages 1–8, 2013.

[131] Michael Wetter, Philip Haves, and Brian Coffey. Building controls virtual test bed. Technical report, Lawrence Berkeley National Laboratory, 2008.

[132] Daniel A Winkler, Ashish Yadav, Claudia Chitu, and Alberto E Cerpa. Office: Optimization framework for improved comfort & efficiency. In *IPSN*, pages 265–276. IEEE, 2020.

[133] Yifan Wu, George Tucker, and Ofir Nachum. Behavior regularized offline reinforcement learning. *arXiv preprint arXiv:1911.11361*, 2019.

[134] Yue Wu, Shuangfei Zhai, Nitish Srivastava, Joshua Susskind, Jian Zhang, Ruslan Salakhutdinov, and Hanlin Goh. Uncertainty weighted actor-critic for offline reinforcement learning. *arXiv preprint arXiv:2105.08140*, 2021.

[135] Lei Yang, Zoltan Nagy, Philippe Goffin, and Arno Schlueter. Reinforcement learning for optimal control of low exergy buildings. *Applied Energy*, 156:577–586, 2015.

[136] Liang Yu, Shuqi Qin, Meng Zhang, Chao Shen, Tao Jiang, and Xiaohong Guan. Deep reinforcement learning for smart building energy management: A survey. *arXiv preprint arXiv:2008.05074*, 2020.

[137] Tianhe Yu, Aviral Kumar, Rafael Rafailov, Aravind Rajeswaran, Sergey Levine, and Chelsea Finn. Combo: Conservative offline model-based policy optimization. *Advances in neural information processing systems*, 34:28954–28967, 2021.

[138] Zishun Yu and Xinhua Zhang. Actor-critic alignment for offline-to-online reinforcement learning. In *International Conference on Machine Learning*, pages 40452–40474. PMLR, 2023.

[139] Chi Zhang, Sanmukh R Kuppannagari, Rajgopal Kannan, and Viktor K Prasanna. Building hvac scheduling using reinforcement learning via neural network based model approximation. In *BuildSys*, pages 287–296, 2019.

[140] Chi Zhang, Sanmukh Rao Kuppannagari, and Viktor K Prasanna. Safe building hvac control via batch reinforcement learning. *IEEE Transactions on Sustainable Computing*, 2022.

[141] Chi Zhang, Sanmukh Rao Kuppannagari, and Viktor K Prasanna. Safe building hvac control via batch reinforcement learning. *IEEE Transactions on Sustainable Computing*, 7(4):923–934, 2022.

[142] Haichao Zhang, We Xu, and Haonan Yu. Policy expansion for bridging offline-to-online reinforcement learning. *arXiv preprint arXiv:2302.00935*, 2023.

[143] Shangtong Zhang and Richard S Sutton. A deeper look at experience replay. *arXiv preprint arXiv:1712.01275*, 2017.

[144] Tianyu Zhang, Gaby Baasch, Omid Ardakanian, and Ralph Evins. On the joint control of multiple building systems with reinforcement learning. 2021.

[145] Zhiang Zhang, Adrian Chong, Yuqi Pan, Chenlu Zhang, and Khee Poh Lam. Whole building energy model for hvac optimal control: A practical framework based on deep reinforcement learning. *Energy and Buildings*, 199:472–490, 2019.

[146] Zhiang Zhang and Khee Poh Lam. Practical implementation and evaluation of deep reinforcement learning control for a radiant heating system. In *BuildSys*, 2018.

[147] Jie Zhao, Bertrand Lasternas, Khee Poh Lam, Ray Yun, and Vivian Loftness. Occupant behavior and schedule modeling for building energy simulation through office appliance power consumption data mining. *Energy and Buildings*, 121:234–243, 2016.

[148] Yi Zhao, Rinu Boney, Alexander Ilin, Juho Kannala, and Joni Pajarinen. Adaptive behavior cloning regularization for stable offline-to-online reinforcement learning. *arXiv preprint arXiv:2210.13846*, 2022.

[149] Han Zheng, Xufang Luo, Pengfei Wei, Xuan Song, Dongsheng Li, and Jing Jiang. Adaptive policy learning for offline-to-online reinforcement learning. *arXiv preprint arXiv:2303.07693*, 2023.

[150] Yuanyang Zhu, Zhi Wang, Chunlin Chen, and Daoyi Dong. Rule-based reinforcement learning for efficient robot navigation with space reduction. *IEEE/ASME Transactions on Mechatronics*, 27(2):846–857, 2021.