Policy Selection for Transfer Learning in the Building Control Domain

by

Aakash Krishna Guruvyaur Sasikumar

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science

Department of Computing Science

University of Alberta

Abstract

The application of reinforcement learning (RL) to the optimal control of building systems has gained traction in recent years as it can reduce building energy consumption and improve human comfort, without requiring the knowledge of the building model. However, existing RL solutions for building control face challenges, such as slow convergence and suboptimal (or unsafe) actions during the training phase that may lead to high energy use or excessive discomfort. Additionally, the transferability of RL policies to different buildings remains a hurdle.

Offline policy selection is a new domain in RL that aims to efficiently select the best policies from a library of policies for a downstream task. Previous works have shown that diversity-induced RL helps generate policies that generalize to unseen environments, often surpassing baselines, even without retraining. This thesis explores various techniques to select a policy from a library of diverse policies to control the heating, ventilation, and air conditioning (HVAC) system of a commercial building. The main contribution of this thesis is an offline policy selection algorithm that can effectively identify the most suitable policy for transfer to an unseen building environment. Furthermore, an investigation into the impact of the offline dataset utilized for evaluation is also conducted, providing valuable insights into the efficacy of the proposed evaluation technique.

The outcomes of this research hold significant implications for energy conservation in the building sector. By enabling the adoption of RL-based control strategies that overcome the limitations of traditional approaches, this work could contribute to significant reductions in energy consumption and carbon emissions. The proposed framework empowers building operators to achieve energy-efficient control of building systems while minimizing occupant discomfort and facilitating the transfer of policies across different buildings.

Preface

This thesis is original work by Aakash Krishna GS, but due to the collaborative nature of this work, "we" is the primary pronoun in this thesis. A majority of this thesis is based on the following journal publication: Aakash Krishna GS, Tianyu Zhang, Omid Ardakanian, and Matthew E. Taylor, "Mitigating an adoption battier of reinforcement learning-based control strategies in buildings", *Energy and Buildings*, 285:112878, 2023. I was partly responsible for the literature review, methodology development, and manuscript writing. The experiments I conducted include the comparison of offline policy evaluation algorithms and warm-starting policies before deploying them in the target building. Tianyu Zhang was responsible for developing the policy clustering algorithm, running other baselines presented in Chapter 4, parts of the literature review, and the methodology. Omid Ardakanian and Matthew E. Taylor provided feedback, guided the project, and helped edit the manuscript.

Designing and creating the policy library was based on the following conference publication: Tianyu Zhang, Aakash Krishna GS, Mohammad Afshari, Pert Musilek, Matthew E. Taylor, and Omid Ardakanian, "Diversity for Transfer in learning-based control of buildings", in *Proceedings of the Thirteenth ACM International Conference of Future Energy Systems*, pages 556-564, 2022. I was partly responsible for defining the diversity loss, generating the plots, literature review, and writing the manuscript. Tianyu Zhang was responsible for developing the diversity loss and training the diverse policy library, which is used in this work and this thesis. Mohammad Afshari assisted with the literature review and editing of the manuscript. Petr Musilek, Matthew E. Taylor, and Omid Ardakanian guided the project and helped edit the manuscript.

Acknowledgements

The ideas discussed in this thesis have been developed over the past two years in collaboration with many people, without whom this thesis would not have been possible. Firstly, I thank my supervisor, Dr. Omid Ardakanian, who not only provided guidance and support throughout my graduate studies but also helped me develop as a better writer and researcher. Secondly, I want to thank Tianyu Zhang, for helping me get started with this project, inspiring me to think more critically, and for patiently answering all my silly questions.

I am thankful to my peers in CAB-385 for making the environment stimulating and fun for research. Specifically, I would like to thank Javier Sales-Ortiz, Sanku Kumar Roy, Hossein Nekouyan Jazi, Afia Arin, Xin Yang, Sanju Xaviar, Jihoon Og, and Vidushi Agarwal. I also would like to thank my roommates Kushagra Chandak and Subhojeet Pramanik for our deep conversations and for making Canada a fun place for a newcomer like myself.

Last but not least, I would like to thank my parents for their unconditional love and support.

Contents

List of Tables

List of Figures

List of Algorithms

Acronyms

A2C Advatange Actor-Critic.

AC Actor-Critic.

AHU Air Handling Unit.

AI Artificial Intelligence.

ASHRAE American Society of Heating, Refrigerating and Air-conditioning.

BMS Building Management System.

CQL Conservative Q-Learning.

GK Gaussian Kernel.

GN Gradnorm.

GP Gaussian Process.

HVAC Heating, Ventilation, and Air Conditioning.

IPW Inverse Probability Weighting.

KDE Kernel Density Estimation.

MARL Multi-Agent Reinforcement Learning.

MDP Markov Decision Process.

ML Machine Learning.

MMDP Multi-agent Markov Decision Process.

MPC Model Predictive Control.

NAS Neural Architecture Search.

OPE Offline Policy Evaluation.

PG Policy Gradient.

PID Proportional–Integral–Derivative.

PPO Proximal Policy Optimization.

RBC Rule-Based Controller.

RL Reinforcement Learning.

SNIP Single-shot Network Pruning.

SNIPW Self-Normalized Inverse Probability Weighting.

UCB Upper Confidence Bound.

VAV Variable Air Volume.

ZCP Zero-Cost Proxy.

Chapter 1

Introduction

It is estimated that around one-third of the worldwide final energy consumption is due to commercial buildings [19], and the Heating, Ventilation, and Air Conditioning (HVAC) system is the primary consumer in these buildings, accounting for more than 40% of the energy consumed [8]. The HVAC system is responsible for maintaining a comfortable and healthy environment. Research has shown that optimizing the energy consumption of the HVAC system is much more sustainable and cost-effective than retrofitting or upgrading the HVAC system [45]. In a country like Canada, where the outside temperature can reach -40°C in the winter and 35°C in the summer, the HVAC system consumes even more energy to maintain a comfortable environment for the occupants. In pursuit of sustainability and energy efficiency, optimal control of the building HVAC system emerges as a crucial area of research.

A typical HVAC system contains a number of sensors in the air handling unit and ductwork, and inside every thermal zone. Examples of these sensors are thermostats, pressure sensors, carbon-dioxide sensors, etc. The Building Management System (BMS) logs measurements of these sensors and enables the control of supply air temperature, zone temperature setpoints, fan speed, damper position, valve position, etc. By adjusting these control knobs at different times and in various combinations, the HVAC system maintains a comfortable environment for the occupants without wasting too much energy. The HVAC control problem is challenging because it is a multi-objective optimization problem. Specifically, maximizing occupant comfort and minimizing energy consumption are two main conflicting objectives. Balancing these two

objectives is difficult as the occupancy of the building and the heat load on the building changes over time. If the HVAC system is not optimally controlled, it can lead to significant energy wastage, faster wear and tear of equipment, and discomfort for the occupants, to name a few. Traditional HVAC controllers are rule-based controllers with rules being defined based on the intuition of the building manager about building occupancy and heat load. Therefore, they are not optimal, as the rules are not updated dynamically according to how the building is used. In contrast, recent advancements in Machine Learning (ML) and Artificial Intelligence (AI) have opened new avenues for more intelligent and dynamic control strategies. In particular, Reinforcement Learning (RL) has shown significant promise in the building control domain.

Although these methods have potential, their widespread adoption is hindered by several barriers. These barriers include but are not limited to challenges related to safety, generalization, and training costs. Safety is defined as the ability of the controller to operate within the constraints of the building. This is a critical requirement as the controller should not violate the constraints of the building, such as the maximum allowed temperature in a zone. Generalization is the ability of the controller to perform well in new buildings. This is a critical challenge, as each building is unique, and the controller should be able to adapt to the new building without significant training. The training cost is the amount of data required to learn a good control policy. This is a significant challenge as the data requirement for RL is very high, and collecting data from buildings is expensive and time-consuming. In this thesis, we explore the potential of applying RL to the HVAC control domain and address one of the key challenges that hinder its widespread adoption. In particular, we focus on controlling the heating and cooling components of the HVAC system and address the generalization problem by proposing a method to efficiently transfer policies from one building to another.

1.1 HVAC Controls

According to the classical control theory, a controller can be classified into two categories, i.e., open-loop and closed-loop controllers. Open-loop controllers, such as feedforward controllers, do not use feedback from the system to determine the control action, so the action is independent of the system output.

These controllers are efficient when the relationship between the control action and state can be accurately modeled. On the other hand, closed-loop controllers, also known as feedback controllers, use feedback from the system to determine the control action. For example, Proportional–Integral–Derivative (PID) controllers [15] are feedback controllers that incorporate the error, i.e., the difference between the setpoint and current value of a physical quantity, to determine the control action.

In building HVAC control, controllers are roughly classified into three categories, i.e., Rule-Based Controller (RBC), model-based controllers, and learning-based controllers. Generally, rule-based controllers fall under the open-loop category, and the other two fall under the closed-loop category. Historically, RBC controllers have been the first choice for HVAC control applications where the setpoints are chosen by the RBC, and a PID controller is used within each zone to maintain the setpoint. The setpoints for these controllers are determined by the facilities manager. The manual tuning of each setpoint is labor intensive; hence, simple rules are used to determine the setpoints. For example, the setpoint for the heating system is set to 21°C, and the setpoint for the cooling system is set to 24°C. The PID controllers then use these setpoints to determine the control action. These controllers are suboptimal as they are not tailored to the specific building and are often tuned manually. Moreover, these controllers do not take into account the occupancy of the building [61], which can lead to significant energy wastage.

In model-based HVAC control, models of the building are used to determine the control actions. Specifically, models that capture heat transfer, outside temperature, occupancy, and other dynamics can be used to predict the future states of the building. These models can be built based on physics or learned from data. Model Predictive Control (MPC) predicts the future states of a building using these models and determines the control action that minimizes energy consumption while ensuring a comfortable environment. MPC has been shown to be effective in reducing energy consumption [44, 53, 58]; however, building accurate models is challenging as it requires a substantial amount of data or significant domain expertise. Moreover, the data-driven or physics-based models cannot be easily transferred to new buildings as the physical properties and dynamics change significantly as each commercial building is unique. Due to these challenges, MPC has yet to be widely adopted in the

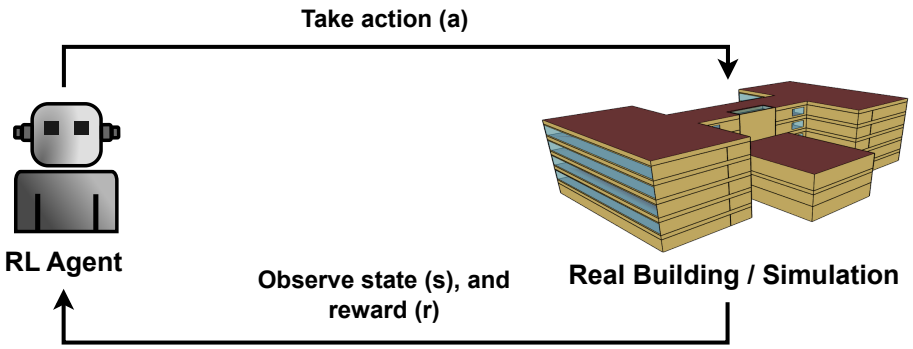


Figure 1.1: Depiction of how the RL agent interacts and learns from the environment.

building control industry.

1.2 Reinforcement Learning-Based Control of the HVAC System

Reinforcement Learning (RL) is a branch of machine learning that has shown significant promise in games, robotics, and continuous control tasks, such as HVAC control [33]. RL aims to learn a control policy (strategy) that maximizes the cumulative reward obtained by interacting with the environment. Figure 1.1 depicts the interaction between the RL agent and the environment. In the HVAC control domain, the environment is either an actual building or its digital twin. The RL agent interacts with the environment by taking an action, and the environment responds with a reward and a new state. The RL agent uses this information to learn a control policy that maximizes the cumulative reward. RL has been shown to learn control policies better than human experts in domains other than building control. For example, RL has beaten the world champion in the game of Go [48] and has been shown to learn control policies for robotics tasks that are better than human experts [30,31]. They are also used in other domains, such as autonomous vehicles, recommendation systems, aligning large language models, and many more. RL approaches can be classified into two categories, i.e., model-free and model-based. In model-free RL, the agent learns the control policy by interacting with the environment without building a model of the environment. In model-based RL, the agent

	RBC	MPC	RL
Main objective	Execute a set of rules	Minimize cost given predicted future states	Maximize cumulative reward
Domain expertise	Requires some expertise	Requires significant expertise	Little expertise(for reward engineering)
Data requirement	None	Order of weeks to months	Order of months to years
Building constraint enforcement	Yes with minor violations	Yes with minor violations	Violations are expected
Building models	None	Requires highly accurate models	No (if model-free); Yes (if model-based)
Transferability	Yes	No	No
Adoption in industry	High	Low	Very low

Table 1.1: Comparison of different control methods.

learns a model of the environment and uses it to determine the control policy.

1.2.1 Challenges of Applying RL to HVAC Control

RL-based controllers have been shown to work well in the HVAC control domain [4, 21, 55, 65]. However, several challenges hinder the adoption of RL algorithms. Firstly, since the agent learns via trial and error, the sample efficiency of RL is very low, requiring several months or even years of interaction to learn a near-optimal control policy. This issue can be circumvented if a high-fidelity simulator is present, but this is not the case in most situations. Training RL policies on an inadequate simulator may lead to poorly performing policies when deployed to real buildings. Secondly, if the number of thermal zones in the building is large, the state representation becomes large, further increasing the data requirement. Thirdly, reward shaping is another challenge with RL-based HVAC control. Careful consideration is required to design the reward signal as the reward signal needs to capture the trade-off between minimizing energy use and maintaining thermal comfort. Lastly, rewards are often delayed in building environments, making learning more difficult.

Table 1.1 compares the control methods with respect to their data requirement, domain expertise, and other factors. Although MPC and RL methods significantly reduce energy consumption and maintain thermal comfort, there are several obstacles preventing their widespread adoption. Specifically, MPC requires significant domain expertise to build accurate heat dynamics models (e.g., the resistance-capacitance network) of the building. RL can overcome this limitation as it requires little domain expertise. However, RL requires significant data to learn a control policy, which may not be readily available

from all buildings. MPCs are slightly better in terms of the data requirement, as they only require some data initially to identify the parameters of the model. Offline RL methods have been shown to reduce the data requirement [5, 26, 46, 63]. In offline RL, the agent learns a control policy using data collected from the environment first; then, the learned policy is deployed. This method reduces the initial training cost of policies. Since offline RL is less sample efficient than online RL, much more data is required to learn a good policy.

Even if the training cost can be reduced to some extent, a significant impediment for all the controllers discussed previously is that they are not transferable to new buildings. Since each building is unique, a physics model or a control policy learned for one building cannot readily transfer to new buildings. A solution to this problem is to use transfer learning. In the building control domain, transfer learning can transfer knowledge from one building to another. Typically in RL, a policy learned in one building can be transferred to a new building and fine-tuned to the new building. This fine-tuning of policies is also known as domain adaptation. Although previous works [24, 42, 59] have shown that transfer learning can be used in the building control domain to reduce the training cost, domain adaptation still requires significant re-training that may not feasible in the real world.

Another aspect of transferability for RL policies is that the policy cannot be transferred if there is a mismatch in the number of thermal zones between the source and target buildings. For example, a policy trained on a building with five thermal zones cannot be transferred to a building with fifteen thermal zones. To address this issue, Multi-Agent Reinforcement Learning (MARL) based controllers have proven to be beneficial [13, 38, 54, 67]. They are energy-efficient and amenable to transfer learning in the building control domain. In MARL, each agent controls a thermal zone in the building, and the agents can learn to coordinate with each other to minimize energy consumption while maintaining thermal comfort. This approach reduces the state-action space, enabling the transfer of policies from one building to another regardless of the number of thermal zones.

Although transfer learning with MARL addresses the generalization problem to some extent, it does not solve the problem. The policies may not transfer well to the new building if the source and target buildings differ sig-

nificantly. This mismatch may incur higher training time during domain adaptation. Previous work has shown that inducing diversity during training can lead to more generalizable policies [11,32,34]. Diversity encourages policies to explore actions the other policies might not have taken, leading to a diverse set of learned skills. This results in policies that may be sub-optimal in the training environment but perform well in novel environments.

1.3 Aim and Scope

This thesis explores how RL can be used for HVAC control and addresses the transferability and generalization issues that hinder its widespread adoption. By taking advantage of the recent advances in policy diversity, RL, and transfer learning for building control, we propose a novel policy selection framework that can select the best policy from a library of diverse policies for new and unseen building environments. The main question this thesis addresses is:

> How to select near-optimal control policies for a new environment from a library of learned policies with as little data as possible?

Our contributions are as follows:

1. We introduce a novel offline off-policy policy selection algorithm to select the best candidate policy from a library of diverse policies learned through interaction with another building.
2. We show empirically that the proposed policy selection algorithm can select the best policy with high probability under certain conditions.
3. We explore the limitations of the proposed algorithm and offer some solutions to overcome them.
4. We make our implementation open source, allowing others to reproduce the result and extend this line of research [1]

[1] The GitHub repository can be found at https://github.com/sustainable-computing/building-MARL-SB3

1.4 Outline

The rest of the thesis is organized as follows. In Chapter 2, preliminaries such as RL, MARL, diversity-induced RL, and building control are discussed. In Chapter 3, we formulate the problem setting and introduce the test building environments used in this thesis. In Chapter 4, we dive deeper into offline off-policy policy selection and discuss an offline policy selection algorithm to pick the best policy to transfer to a new environment. The proposed methodology is tested on the three different building environments with varying weather profiles introduced in Chapter 3. In Chapter 5, we explore the limitations of our proposed methodology and offer some solutions. Finally, Chapter 6 summarizes the work, discusses its limitations, and details future directions.

Chapter 2

Background

This chapter discusses the background required for understanding the contribution of this thesis. We start with an overview of the Markov decision process framework and RL concepts such as offline policy evaluation, policy selection, diversity-induced RL, and different RL algorithms. We then survey the related work on building HVAC control introduced in 1.1.

2.1 Reinforcement Learning

This section provides a brief overview of Reinforcement Learning followed by the notion of policy diversity in Reinforcement Learning. We then discuss Offline Policy Evaluation (OPE) and prior work in this area. Lastly, we discuss policy selection and review recent literature in this area.

2.1.1 Overview

Reinforcement Learning (RL) [49] is a learning framework that allows agents to optimize their behavior in an environment through trial and error. In general, the agent's goal is to maximize its cumulative reward. The Markov Decision Process (MDP) is a mathematical framework for modeling sequential decision-making problems. A few assumptions need to be satisfied for a problem to be modeled as an MDP. First, the problem needs to be sequential, i.e., the current state of the environment must be dependent on the previous state. Secondly, the environment has to be Markovian, meaning that the current state of the

environment needs to contain all the relevant information of the past, so that the next state is independent of the past states given the current state.

An MDP is defined by the tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma \rangle$. The state space $\mathcal{S}$ is the set of all possible states of the environment. The action space $\mathcal{A}$ is the set of the agent's possible actions. Once an action is taken, the agent observes a new state $s' \in \mathcal{S}$, where the transition probability from state s to state s' is given by the transition function $\mathcal{T}(s, a, s') = \Pr(s'|s, a)$. The reward function $\mathcal{R} : \mathcal{S} \times \mathcal{A} \Rightarrow \mathbb{R}$ defines the reward r received by the agent when it takes action a in state s. The discount factor γ is a hyperparameter that controls the importance of future rewards. If γ is close to 0, the agent will only care about the immediate reward; if γ is close to 1, the agent will also care about future rewards.

Irrespective of the RL algorithm used to train the agent, the common goal of an RL agent is to maximize its return which is defined as the discounted sum of rewards or $G_t = \sum_{k=1}^{\infty} \gamma^k r_{t+k}$ where $\gamma \in [0, 1]$. This is done by learning a policy $\pi : \mathcal{S} \to \mathcal{A}$ that maps the state of the environment to an action taken by the agent. Generally, the policy is parametrized by learnable parameters $\theta \in \Theta$ and denoted by π_θ. The objective then is to find the optimal parameters θ^* that maximize the expected return G_t or $\theta^* = \text{argmax}_{\theta \in \Theta} \mathbb{E}_{\pi_\theta}[G_t]$.

2.1.2 Types of Reinforcement Learning Algorithms

RL algorithms are classified as model-based and model-free. Model-based algorithms learn the transition function $\mathcal{T}$ and the reward function $\mathcal{R}$ of the environment. In other words, the agent also learns to model the environment in addition to learning a policy. The model is often used to simulate the environment and generate trajectories to plan the agent's actions ahead. The Dyna-Q algorithm [49] is an example of a model-based RL algorithm. There have been several recent advances in model-based RL. For example, DreamerV3 [17] is a state-of-the-art model-based RL algorithm that uses world models to learn policies. It has been shown to beat other models in several continuous control tasks, such as the DeepMind Control Suite [51] and Minecraft. A significant advantage of model-based RL algorithms is that they can learn policies with fewer samples than model-free RL algorithms, i.e., they are more sample efficient.

On the other hand, model-free RL algorithms do not learn a model of the environment. Instead, they learn a policy by interacting with and observing the rewards received from the environment. Model-free RL algorithms are further classified into value-based and policy gradient-based algorithms. Value-based algorithms learn the state-value function $V^{\pi}(s)$ or the action-value function $Q^{\pi}(s,a)$ of the policy π. The state-value function $V^{\pi}(s)$ is the expected return starting from state s and following policy π after that. The action-value function $Q^{\pi}(s,a)$ is the expected return starting from state s, taking action a and following policy π thereafter.

$$V^{\pi}(s) = \mathbb{E}_{\pi}\left[G_t | s_t = s\right],$$
$$Q^{\pi}(s,a) = \mathbb{E}_{\pi}\left[G_t | s_t = s, a_t = a\right]$$

By learning the value function, the agent can select the action that maximizes the value function in a given state. The Q-learning algorithm [49,56] and Deep-Q learning [37], where a neural network is used to parameterize the action-value function, are examples of a value-based RL algorithm. A significant disadvantage of Q-learning is that it is unsuitable for continuous action and state spaces, as Q-learning algorithms fundamentally rely on discrete state and action spaces.

Policy Gradient (PG)-based algorithms, on the other hand, learn the policy directly. Policy gradient algorithms can model the policy's stochasticity and are suitable for continuous action and state spaces. Due to PG-based algorithm's ability to model stochastic policies, the action preferences change smoothly over time as compared to value-based algorithms, where the action preferences can change abruptly. This makes PG-based algorithms have better convergence properties than value-based algorithms. The policy gradient theorem [49] states that the gradient of the expected return with respect to the policy parameters is given by:

$$\nabla_{\theta} J(\theta) = \mathbb{E}_{\pi_{\theta}}\left[\nabla_{\theta} \log \pi_{\theta}(s,a) Q^{\pi_{\theta}}(s,a)\right],$$

where $J(\theta)$ is the expected return of the policy π_{θ}, and θ are the parameters of the policy. The policy gradient theorem is the basis for PG-based RL algorithms. Neural networks are most commonly used to parameterize the policy, and the policy is learned by stochastic gradient descent. A disadvantage of

PG-based algorithms is that due to gradient descent being a local optimization method, they can get stuck in local optima. Whereas, value-based algorithms are global optimization methods and are less likely to get stuck in local optima. The REINFORCE algorithm [49] is an example of a policy gradient-based RL algorithm.

Actor-Critic (AC) algorithms are a combination of value-based and PG-based algorithms. They incorporate two neural networks into the algorithm, one to learn the policy and the other to learn the state-value function. The policy network is called the *actor*, and the state-value network is called the *critic*. This approach has been shown to reduce variance in the policy gradient estimates and improve the algorithm's stability. Advatange Actor-Critic (A2C) [36] and Proximal Policy Optimization (PPO) [47] are examples of actor-critic algorithms. In this thesis, we restrict our focus to actor-critic methods, specifically, the PPO algorithm. PPO is more stable than other approaches and can be used for continuous control problems. PPO has been shown to work well in several domains [62]. The loss of PPO is given by:

$$L_{PPO} \doteq \hat{\mathbb{E}}\left[\min(\rho(s_t, a_t)\hat{A}_t, \text{clip}(\rho(s_t, a_t), 1-\epsilon, 1+\epsilon)\hat{A}_t)\right], \tag{2.1}$$

where $\hat{A}_t$ is the estimated advantage at time t, $\rho(s_t, a_t)$ is the ratio of the probability under the new and old policies, respectively, and *clip* projects this probability ratio onto $[1-\epsilon, 1+\epsilon]$ so it cannot be too far away from 1. The advantage estimate can be calculated from $\hat{A}_t \doteq G_t'^{\pi_{\text{old}}} - V^{\pi_{\text{old}}}(s_t)$, where $G_t'^{\pi_{\text{old}}}$ is the discounted reward starting from s_t and running π_{old} for a fixed number of timesteps, and $V^{\pi_{\text{old}}}$ is the state-value function under π_{old}. The ratio $\rho(s, a)$, also known as the *importance sampling ratio*, is defined as $\frac{\pi(a_t|s_t)}{\pi_{old}(a_t|s_t)}$ with π being the learning policy and π_{old} representing the old policy. Finally, ϵ is the hyperparameter controlling the size of the updates by constraining them to a trust region.

2.1.3 Multi-Agent Reinforcement Learning

So far, we have discussed single-agent RL algorithms. In Multi-Agent Reinforcement Learning (MARL), multiple agents simultaneously interact with the environment and learn a policy. This setting is of interest because, in a building, there may be multiple thermal zones, each with its own local con-

trol system e.g., the Variable Air Volume (VAV) system. MARL allows us to train a policy for each zone separately and then transfer the policies to a new building. Multi-agent Markov Decision Process (MMDP) is a generalization of MDPs to multiple agents. This framework is used to model MARL problems. In MMDPs, each agent has its own state space, action space, and reward function. MMDPs are defined by the tuple $\langle \mathcal{N}, \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma \rangle$. $\mathcal{N}$ is the set of all agents, $\mathcal{S}$ is the joint state space, $\mathcal{A}$ is the joint action space, $\mathcal{T}$ is the transition function, $\mathcal{R}$ is the joint reward function, and γ is the discount factor.

2.1.4 Further Classifications

On-Policy and Off-Policy Algorithms

RL algorithms can be further classified into *on-policy* and *off-policy* algorithms. The main difference between on-policy and off-policy algorithms is that on-policy algorithms use the current policy to collect data. In contrast, off-policy algorithms use a different policy to collect data. In off-policy algorithms, the policy used to collect data is called the *behavior policy*, and the policy being learned is called the *target policy*. In on-policy algorithms, the behavior and the target policies are the same.

Off-policy algorithms are vital because they can learn from data produced by other policies. In other words, off-policy algorithms can learn from data collected by a human expert or a hand-made policy or a different RL algorithm. Another advantage of off-policy algorithms is that they can learn from parallel data collection. Some examples of off-policy algorithms are Q-learning [56] and Deep Q-learning [37].

On-policy algorithms are generally more stable and easier to implement. In settings where the agent can explore freely, on-policy algorithms are preferred. Some examples of on-policy algorithms are REINFORCE [49] and PPO [47].

Online and Offline Algorithms

Depending on the problem setting, RL algorithms can be classified into online and offline algorithms. In online RL, the agent interacts with the environment

and learns a policy simultaneously. In offline RL, also known as batch RL, the agent learns a policy from a static dataset of trajectories collected from the environment. In offline RL, the agent cannot explore the environment and collect new data, as the dataset is static. Batch Constrained Q-learning (BCQ) [14] is an example of an offline RL algorithm.

Episodic and Continuing Tasks

RL tasks can be further classified into episodic and continuing tasks. Episodic tasks are the tasks that have a terminal state. In RL, the terminal state is the state where the episode ends. For example, in the game of chess, the terminal state is when one of the players wins. Once the game is over, the agent is reset to the initial state irrespective of the position of the agent in the previous episode. In other words, each episode is independent of each other.

On the other hand, continuing tasks are the tasks that do not have a terminal state. Stock trading is an example of such a task. In this task, there is no terminal state and the agent is not reset to the initial state after each trading period.

2.1.5 Policy Diversity

In general, RL algorithms are designed to find a single near-optimal policy. However, when this policy is often transferred to a new unseen environment, it often does not perform well. Incorporating diversity when learning policies is a promising approach to address this issue of generalization. McKee et al. [35] survey different types of diversity and how they affect learning in a multi-agent system and classify diversity broadly into two types: *policy diversity* and *environmental diversity*. In environmental diversity, the agents are trained on variants of the environment. This approach aims to make the agents generalize to a broader class of environments [20, 35].

On the other hand, policy diversity focuses on finding several distinct sub-optimal policies in the training environment. Masood et al. [32] use maximum mean discrepancy to encourage diversity in the policies learned by the agents. Eysenbach et al. [11] propose a method where the agents do not need rewards at all. Instead, by just incorporating diversity, the agents are able to learn

to solve a set of complex continuous control tasks. The goal of diversity-induced RL is to obtain a diverse collection of skills learned from one or more environments. These skills can then be transferred to a new environment to solve a similar but different task. The tasks can be different in terms of the dynamics of the environment, but the tasks have to be similar in terms of the goal of the agent. For example, a diverse set of skill learnt on robot locomotion tasks cannot be transferred to the building control task.

In this thesis, we follow the approach of [66] for incorporating diversity in RL. This paper proposes an iterative diversity-induced RL algorithm for building HVAC control, which entails modifying the PPO loss expressed in Equation 2.1 to incorporate diversity. The new loss function is as follows:

$$L' = L_{PPO} + wL_{diversity},$$

where L_{PPO} is the standard PPO loss, $L_{diversity}$ is the diversity loss, and w is the weight of the diversity loss. $L_{diversity}$ is defined as follows:

$$L_{diversity} = -\frac{\sum_{\pi' \in \Pi_{learned}} \sum_{(s,a) \in \text{exp}} \frac{max\left(\frac{\max(\pi(a|s),\pi'(a|s))}{\min(\pi(a|s),\pi'(a|s))}, \bar{\rho}\right)}{\left|G^{\text{exp}}(s) - V^{\pi'}(s)\right|}}{|\Pi_{learned}|},$$

where $\Pi_{learned}$ is the set of learned policies, exp is the dataset of trajectories collected from the environment, $\bar{\rho}$ is the maximum importance sampling ratio, $G^{\text{exp}}(s)$ is the return of the expert policy, and $V^{\pi'}(s)$ is the value of the state s under the policy π'. The term $\left|G^{\text{exp}}(s) - V^{\pi'}(s)\right|$ measures the estimation bias of a learned policy given the current trajectory. The term $\frac{\max(\pi(a|s),\pi'(a|s))}{\min(\pi(a|s),\pi'(a|s))}$ measures the difference between the learned policy and the expert policy. When $L_{diversity}$ is small, the behavior policy is different from the learned policies, and when $L_{diversity}$ is large, the behavior policy is similar to the learned policies. The diversity weight w dictates how much importance is to be given to diversity. A very high value of w will result in a more diverse set of policies, but it will sacrifice optimality and may yield useless policies. When w is set to 0, the algorithm reduces to standard PPO, where the goal is to find a single near-optimal policy.

2.1.6 Offline Off-Policy Evaluation

In this section, we discuss Offline Policy Evaluation (OPE) and the different approaches to OPE. OPE is the problem of estimating the value of a policy using a static dataset of trajectories collected from a different policy. For example, in the building HVAC control problem, the operational log data collected under the default rule-based controller's can be used to evaluate a given policy.

The policy whose value is being estimated is called the *evaluation policy*, and the policy that collected the dataset is called the *behavior policy*. Let the historical dataset be denoted by $\mathcal{D} = \{(s_t, a_t, r_t)_{t=1}^n\}$ where s_t, a_t, r_t are the state, the action taken, and the reward received at time t, respectively. The goal of OPE is to estimate the value of the evaluation policy π_e using the dataset $\mathcal{D}$. The most popular OPE methods are based on importance sampling, examples of which are Inverse Probability Weighting (IPW) [43] and Self-Normalized Inverse Probability Weighting (SNIPW) [50]. In general, SNIPW is shown to be more stable in certain tasks as the support of the rewards bounds its value, and its variance is smaller than IPW [22]. Given the evaluation policy π_e and the behavior policy π_b, the value of the π_e under IPW and SNIPW is defined as follows:

$$\hat{V}_{\text{IPW}}(\pi_e; \mathcal{D}) \doteq \frac{1}{n} \sum_{t=1}^{n} \rho(s_t, a_t) r_t,$$
$$\hat{V}_{\text{SNIPW}}(\pi_e; \mathcal{D}) \doteq \frac{\sum_{t=1}^{n} \rho(s_t, a_t) r_t}{\sum_{t=1}^{n} \rho(s_t, a_t)},$$

where $\rho(s, a) \doteq \frac{\pi_e(a|s)}{\pi_b(a|s)}$ is the importance sample ratio, $\mathcal{D}$ denotes the offline dataset from which the trajectory was sampled, and s_t, a_t and r_t respectively represent the state, action taken, and reward received at time step t. The OPE methods mentioned previously assume that actions are discrete and use rejection sampling to filter the dataset. However, this approach cannot be extended to work with continuous actions as rejection sampling does not work in the continuous setting [23]. To overcome this limitation, Kallus et al. [23] employ Kernel Density Estimation (KDE) to calculate the value of a policy,

which is given by:

$$\hat{V}_{\text{Kernel}}(\pi_e; \mathcal{D}) \doteq \mathbb{E}\left[\frac{1}{h} K\left(\frac{\text{argmax}_{a'_t}\, \pi_e(a'_t|s_t) - a_t}{h}\right) \frac{r_t}{\pi_b(a_t|s_t)}\right].$$

Here K is the kernel function, such as the Gaussian kernel, and h is the bandwidth which is a hyperparameter. When a Gaussian kernel is adopted, we refer to this method as GK.

2.1.7 Policy Selection

Policy selection refers to the process of selecting the best policy from a set of policies for the agent to follow to maximize its cumulative reward. Policy selection can be divided into two categories: online policy selection and offline policy selection. In online policy selection, the policies are executed and evaluated on the environment, and the best policy is selected based on their performance on the environment. In offline policy selection, the policies are evaluated on a dataset of trajectories collected from the environment. The dataset of trajectories is obtained from the behavior policy π_b.

Online policy selection algorithms include Upper Confidence Bound (UCB) [3]. Under the UCB algorithm, the policies are treated as arms in a multi-armed bandit problem. At each time step, the policy with the maximum upper confidence bound is selected and executed in the environment. The upper confidence bound for a policy i is given by $\bar{r}_i + \sqrt{\frac{2 \log t}{n_i}}$ where $\bar{r}_i$ is the average reward obtained from policy i, t is the current time step, and n_i is the number of times policy i has been executed. As we will discuss in Chapter 4, this approach cannot be readily adopted in the building control domain.

Offline policy selection, however, is more suitable for the building control domain and is a relatively new sub-domain in offline reinforcement learning. References [25, 41, 60] are some of the recent works in this area. Konyushova et al. [25] propose a method for offline policy selection named *Active Offline Policy Selection* where all the policies in the policy library are evaluated via OPE first. Then a Gaussian Process (GP) with a UCB acquisition function is employed to select the best policy by evaluating them on the environment in an online fashion. Paine et al. [41] evaluate policies via Fitted-Q Evaluation [28] and then select the policy with the best estimated value for deployment. Yang

et al. [60] describe a method for policy selection where instead of a point estimate of the value of a policy, a distribution over the policy's value is obtained. The policy selection algorithm can change based on the downstream tasks after a distribution is estimated. Depending on the downstream task, the selection algorithm may select policies with the least variance and highest minimum value, to name a few.

In essence, the offline policy selection problem concerns assigning a rank $\mathcal{O} \in \text{Perm}([1, N])$ to the policies in the policy library $\{\pi_i\}_{i=1}^{N}$ given an offline dataset $\mathcal{D}$. Each policy in the policy library has an unknown value μ_i and an estimated policy value $\hat{\mu}_i$, which is calculated from the offline dataset. Different approaches for policy selection differ in how the value estimates are calculated and how these affect the rank assignment.

Evaluation Metrics

To compare the performance of different ranking methods, we first need to estimate each policy's expected return in the policy library by running it in the target environment. This process yields the *ground truth ranking* of the policy library. We refer to the expected return of each policy as its *actual value*. The two main metrics used to evaluate the ranking and selection performance of policy selection algorithms are the *Spearman's rank correlation* and the *Regret@k*. To calculate Spearman's rank correlation, we calculate the Pearson correlation between the ground truth ranks and the estimated policy ranks. The higher the correlation coefficient, the closer the estimated rank set is to the ground truth set. Regret@k is the difference between the actual value of the best policy in the ground truth set and the actual value of the best policy in the estimated top-k set. The lower the value of Regret@k, the better the performance of the policy selection algorithm.

2.2 Neural Architecture Search

Neural Architecture Search (NAS) has become the standard approach in deep learning to discover the best neural networks among a set of candidate architectures for a given supervised learning task. Since the search space of neural architectures can be extremely large, manual evaluation of all possible

architectures is infeasible. Devising lightweight evaluation methods has been a primary focus of NAS research [18]. Below we discuss different approaches to lightweight evaluation of neural architectures.

2.2.1 Zero-Cost NAS

Efforts have been made to identify low-cost or Zero-Cost Proxy (ZCP) tasks to rank neural networks at initialization (i.e., before training) [1, 29]. In general, ZCP tasks can be divided into two categories, gradient-based and gradient-free. Gradient-based methods use a mini-batch of data to compute the gradient of the loss function with respect to the network parameters. These gradients are aggregated into a single scalar value, which is used to rank the neural architectures. Gradient-based methods differ in how the gradients are aggregated, but all use the aggregate as a heuristic to predict how the neural network would perform in a task. Gradient-free methods, on the other hand, do not use a mini-batch of data to calculate the gradients of the loss. They instead calculate a scalar value of the weights of the neural network at initialization in order to predict how they might perform in the downstream supervised learning task. In practice, we found that gradient-free methods perform worse than gradient-based methods. Hence, we will only discuss gradient-based methods in this thesis.

Lee et al. [29] introduce a saliency metric, Single-shot Network Pruning (SNIP), that approximates the change in loss when a connection is removed. This helps identify connections in the neural network important to the given task before training the network using a mini-batch of data. While SNIP was initially proposed for network pruning, it can be used as a proxy for NAS based on the observation that a neural network that attains a higher SNIP will perform better in a given task [1]. SNIP is calculated as follows:

$$\mathcal{S}_{\text{SNIP}} \doteq \left| \frac{\partial \mathcal{L}}{\partial \theta} \odot \theta \right|,$$

where $\mathcal{L}$ is the loss function of the neural network with parameters θ, and $\odot$ denotes the Hadamard product.

Abdelfattah et al. [1] empirically evaluate various ZCP metrics to compare their efficiency in ranking neural networks. They also propose a new metric,

called Gradnorm (GN), which can be used for NAS, and is defined as the sum of the Euclidean norm of the gradients after back-propagating the loss computed from a mini-batch of data.

In deep RL, the policy is represented by a neural network. This policy π_θ parameterized by a neural network predicts the action to be taken given the state, which is the input. Since ZCPs successfully predict the performance of neural networks in supervised learning tasks, we conjecture that they can also be used for ranking policies. However, in RL, the policy must be run on the target environment to calculate the loss. Since we focus on the problem of offline policy selection, this constraint is prohibitive. In the building control domain, it is not feasible to deploy all the control policies in the policy library on the target building to assess their performance. In this thesis, we propose a modification to the ZCPs proposed in [1, 29] to make them work in the RL setting. Specifically, we use importance sampling mentioned in Section 2.1.6 with ZCPs. We explain this method in detail later in Section 4.2.

2.3 Previous Work on HVAC Control

Building control is the process of controlling the various systems in a building to achieve a desired goal. In this thesis, we restrict the scope of building control to controlling the HVAC system of a building to reduce energy consumption. In this section, we discuss how standard rule-based controllers work, followed by a discussion on Model Predictive Control (MPC).

2.3.1 Rule Based Controllers

Conventionally, building HVAC control is predominantly rule-based and heuristic based on the building operator's experience. Primarily, rule-based controllers rely on predefined temperature setpoints and PID controllers to maintain these setpoints. Building control experts, the American Society of Heating, Refrigerating and Air-conditioning (ASHRAE), have developed a set of recommendations [2] for building HVAC control. These methods are widely adopted but are predetermined and not tailored to the specifics of buildings.

Most HVAC control strategies under rule-based controllers only realize lim-

ited energy savings. However, by adopting approaches discussed in later sections, such as MPC and RL, some case studies have shown that we can achieve energy savings from 15% to 30% [7, 9].

2.3.2 Model Predictive Control

Drgoňa et al. [9] have surveyed the most notable Model Predictive Control (MPC) methods used in the building HVAC control domain. The MPCs use models of different building characteristics and predict future values from them For example, some of the models used by MPC are the heat transfer model of the building, weather forecasts, and occupancy forecasts. By looking at these projections, the MPC can optimally choose control actions that will minimize the energy consumption of the building.

The main bottleneck for MPCs is the models used to predict the future states of the building. Complex models increase the computational cost of the MPC, and inaccurate models can lead to sub-optimal control actions. There are three main paradigms used to build these models. The first is the *white-box* approach, where the model is built from the principles of heat transfer, conservation of energy, and mass. The geometry of the building, materials used, and equipment specifications are used to calculate the model's parameters. White-box models are accurate but challenging to build because of the required significant domain and building knowledge.

The second approach is the *grey-box* approach, where the model is built from a combination of simplified physical models and data-driven methods. The equations used for models in the grey-box approach are generally more straightforward than the white-box approach, leading to lower computational costs. The parameters of the model are estimated from data collected from the building. The parameter estimation methods can be divided into batch and online methods. In batch methods, the parameters are estimated by minimizing the estimation error over a fixed time period. Online methods, on the other hand, estimate the parameters by minimizing the estimation error at each time step.

The third approach is the *black-box* approach, where the model is built using purely data-driven methods. Black-box approaches have lower developmental costs since they do not require any physical relationships of the building

and do not make any assumptions about the building. However, they require much more data to train and are less accurate when predicting outside the training distribution.

MPCs can also handle constraints on the control actions and the state of the building. MPC is a constrained optimization problem that estimates the optimal control actions by minimizing a cost function over a finite time horizon. The heat transfer model of the building, along with predictive models for weather and occupancy, are typically used to solve this finite-horizon optimization problem. Once the solution is found, the first control action is implemented, new observations are made for the next time slot, and the optimization problem is solved again.

2.3.3 RL-Based Controllers

Challenges such as sensor noise, disturbances, delay in control actions, and varying preferences of occupants hinder the adoption of RL methods in real-world building systems [40]. To address these challenges, offline RL methods have emerged as a promising approach. Dey et al. [7] propose an imitation learning approach to learn a policy. By employing imitation learning on a static dataset collected from a building, they can reduce the training cost of the RL algorithm. In a similar vein, Schepers et al. [46] employ the Conservative Q-Learning (CQL) algorithm [26] to learn a policy from a static dataset. Although these methods were evaluated using simulators, recent works that have shown that these methods can be deployed in real-world buildings [27, 39]. MARL has also shown promise in energy-efficient control of building HVAC [13, 54]. It enables controlling different knobs in one or multiple building systems. For example, Zhao et al. [67] use separate agents to manage electricity flow, cooling components, and heating components in a building.

Transfer learning is also used to reduce training costs for the building control problem. Often a digital twin of the building is used to train a policy; then, the policy is transferred to the real building. Pinto et al. [42] survey the different transfer learning methods used in the building control domain. One of the main challenges that hinder transfer learning is preventing negative transfer. Negative transfer occurs when the policy learned in the source

building performs worse than the default rule-based controller in the target building. In other words, the policy learned in the source building cannot generalize to the target building. Xu et al. [59] propose a method to prevent negative transfer using a novel approach. Their approach involves decomposing the policy neural network into a transferable front-end network and a trainable back-end network. The front-end network captures the building-agnostic behavior, whereas the back-end network needs to be trained on the target building. Although this approach somewhat reduces the training cost of RL, control performance can still be poor while the back-end network is being trained. In another line of work, Fazel et al. [24] propose a method to augment the training data collected from the target building. The authors use generative adversarial networks to learn the building performance profile from the actual data and generate synthetic data that reflect climate and operation variations while keeping the building profile the same. However, one year of data is required to train the generative model, which may not be readily available in all buildings.

In our previous work, we showed that introducing diversity when learning policies can improve the performance of the policies in the target building [66]. However, no algorithm was explored to efficiently pick the best-performing policy. In this thesis, we propose a novel method for offline policy selection that employs techniques from neural architecture search.

Chapter 3

Building Control as an RL Problem

In this chapter, we cast the building HVAC control problem as a multi-agent reinforcement learning problem. We describe the state and action spaces of each agent, and the reward function. We then introduce three building environments and the simulator used to evaluated various policy selection methods.

3.1 Problem Formulation

We consider an HVAC system that consists of one or multiple Air Handling Units (AHUs) and Variable Air Volume (VAV) boxes as depicted in Figure 3.1. The optimal HVAC control is modeled as a sequential decision making problem where an agent interacts with the building to control various knobs such as actuators in the VAV systems, and receives a reward in return which is used to learn a control policy. While a single agent can control the entire building, it prevents the policy from being transferred to a new building with a different state-action space. We frame the HVAC control problem in a MARL setting where each agent is responsive for controlling a single zone; several independent agents control a building, each acting in their respective zone. Our Multi-agent Markov Decision Process (MMDP) is a tuple $(N, \mathcal{S}, \mathcal{A}_{i,i\in\{1,\dots,N\}}, \mathcal{R}_{i,i\in\{1,\dots,N\}}, \mathcal{T}, \mathcal{H})$ where, N is the number of agents and $\mathcal{S}$ is the state space where each agent receives readings of six physical or virtual sensors namely, mean temperature ($^{\circ}C$), mean humidity (%), outdoor temperature ($^{\circ}C$), so-

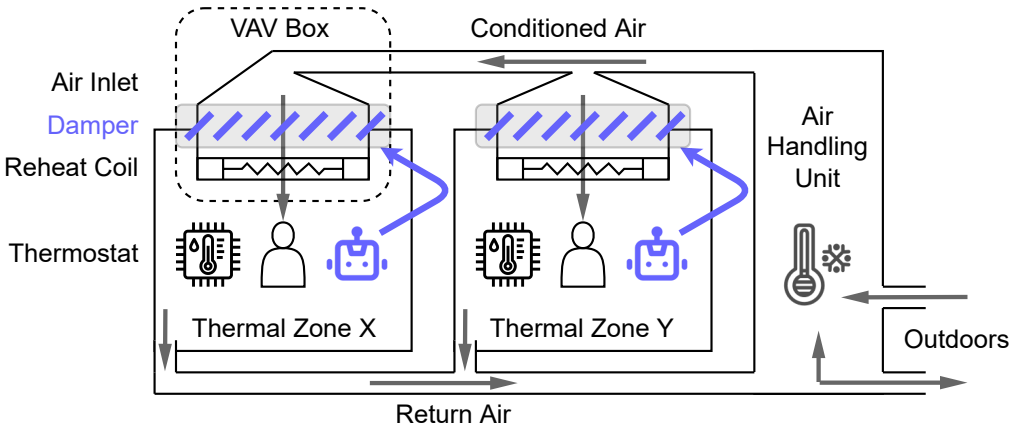


Figure 3.1: Illustration of an air loop in a multi-zone building equipped with a forced-air heating and cooling system.

lar irradiation (W), binary occupancy state, and hour of the day ($0 - 23$). $\mathcal{A}_i$ denotes the action space of each agent i is the minimum position of the damper in its respective VAV box. The minimum damper position is a value in $[0.1, 1]$, where 0 indicates that the damper is fully closed, and 1 indicates that the damper is fully opened. For example, if the minimum damper position is set to 0.2, the damper can be opened anywhere between 20% and 100%. The AHU control points and all other VAV control points are adjusted by the controller in EnergyPlus [6] using the *predictive system energy balance* algorithm [6]. The RL agents do not interfere with this process, ensuring that each zone's thermal comfort requirements are satisfied. $\mathcal{R}_i$ is the reward function for agent i when it takes action a in state s. To incentivize the agents to minimize HVAC energy consumption, we define the reward of each agent as the heating energy consumption of the respective VAV system with a negative sign.

$$r_i(s, a) = -\mathrm{E}_i^{\text{vav\_heating}}(s, a) \tag{3.1}$$

where $\mathrm{E}_i^{\text{vav\_heating}}(s, a)$ is the heating energy consumption of the VAV system in zone i when the agent takes action a in state s. Another reward function that we consider incorporates the cooling energy consumption of the AHU system. This reward signal is the negative of the sum of the cooling energy consumption of the AHU system and the heating energy consumption of the VAV system.

$$r_i(s, a) = -\mathrm{E}^{\text{ahu\_cooling}}(s, a) - \mathrm{E}_i^{\text{vav\_heating}}(s, a) \tag{3.2}$$

Table 3.1: State variables and the action of each agent.

State	Zone mean temperature	$^{\circ}C$
	Zone mean humidity	%
	Zone occupancy	Binary
	Outdoor temperature	$^{\circ}C$
	Solar radiation	W
	Hour of the day	Integer
Action	VAV minimum damper position	%

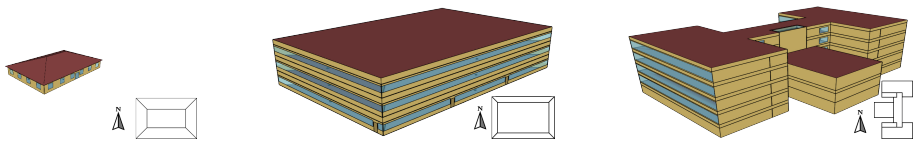
where $\mathrm{E}^{\text{ahu\_cooling}}(s, a)$ is the cooling energy consumption of the AHU system when the agent takes action a in state s. For training the agents, only the first reward function (3.1) is used. Equation (3.2) is used to perform offline evaluation of the learned policies. This approach is discussed in Chapter 5. $\mathcal{T}$ is the transition function that defines the transition probability for transitioning from state s to s' when agent i takes action a. This is defined by the EnergyPlus simulator. $\mathcal{H}$ defines the episode length. We model this problem as an episodic task to evaluate the policies over a fixed period of time. In this thesis, we set the episode length as one month, with 15-minute time steps. Table 3.1 summarizes each agent's state and action variables. In our MARL formulation, each agent aims to learn a policy π_i that maximizes the expected discounted return G_i, given by:

$$G_i \doteq \mathbb{E}\left[\sum_{t=0}^{\mathcal{H}} \gamma^t \mathcal{R}_i(s_t, \underset{a_t \in \mathcal{A}_i}{\operatorname{argmax}}\, \pi_i(a_t|s_t))\right]. \tag{3.3}$$

In this setting, we set the discount factor γ to 1 since the length of each episode is finite. Since agents have different rewards, this is a competitive MARL setting. Although this formulation might increase the convergence time, it makes it possible to train these agents and transfer a subset of them to a new building. This advantage outweighs the drawbacks of slower convergence.

3.2 Building Environments

To study the effects of our proposed policy selection algorithm, we evaluate it using the EnergyPlus simulator [6] on three different building environments, including a real campus building. The COBS framework [64] is used to in-



(a) Building A (b) Building B_{Denver} (c) Building C

Figure 3.2: The 3D view and floor plan of the buildings considered in this paper where north is marked on each floor plan

terface with EnergyPlus and train the agents. Each building has a unique occupancy schedule which is encoded in the EnergyPlus model. We assume that if a control policy outperforms other policies with respect to the HVAC energy use reported by EnergyPlus without degrading thermal comfort, it also outperforms them in the real building.

- **Building A** is a small office prototype building as defined by ASHRAE Standard 90.1 [10]. Figure 3.2a shows the floor plan and 3D model of this building. It contains five thermal zones (4 perimeter zones and 1 core zone) and is located in Denver, Colorado. Each zone is conditioned using a dedicated AHU and contains a VAV system. The total floor area of this building is 511.16 m^2.

- **Building B_{Denver}** is a medium office prototype building as defined by ASHRAE Standard 90.1 [10]. It contains 15 thermal zones across three floors and is located in Denver, Colorado. Figure 3.2b depicts the floor plan of this building. There are 4 perimeter zones and 1 core zone on each floor. Each floor is conditioned using an AHU and all zones are equipped with a VAV system. Its total floor area is 4,982.19 m^2.

- **Building $B_{SanFrancisco}$** is the same building as B_{Denver} with two main differences: 1) it is located in San Francisco, California and 2) its orientation is rotated by 45 degrees (clockwise). We make these changes so as to investigate whether any of the learned policies works well after transfer to a building with a different orientation in a different climate.

- **Building C** is a medium campus building representing the model of the building that houses the Department of Energy Engineering at Sharif University of Technology in Tehran, Iran.[1] It contains 26 thermal zones

[1]Model is downloaded from https://github.com/DOEE-BMS/EnergyPlus-Model

spread across five floors, 11 of which are equipped with a VAV system. The HVAC, lighting, and blind systems are modelled such that they match the design of these systems in the physical building. We assume the building is located in San Francisco, California, because weather data is lacking for its actual location. The total floor area of this building is 5,051 m^2.

Chapter 4

Policy Selection for Controlling a Novel Building

Policy selection is defined as the process of selecting the best policy from a set of policies, such that the chosen policy maximizes the expected return on the target environment. In Section 2.1.7 we discussed the general problem of policy selection and different approaches to solving it. However, when applied to the building control domain, new challenges arise pertaining to the cost of exploration and the number of policies in the policy library. Due to these challenges, the problem becomes non-trivial and requires different solutions. This chapter details the issues related to policy selection in the building control domain and presents a novel policy selection method that is based on zero-cost proxies (ZCPs) discussed in Section 2.2.1. We then conclude by evaluating our proposed methodology on the three building environments introduced in Section 3.2.

4.1 Online vs. Offline Policy Selection in the Building Control Domain

Recall that policy selection is divided into two categories: online policy selection and offline policy selection. Online policy selection requires evaluating the policies on the target environment, and the policy with the highest expected return is selected. This is not very feasible in the building control

domain as exploring sub-optimal policies is costly as it may increase energy consumption and cause discomfort to the occupants. Another issue faced by employing online policy selection is that testing all policies in the policy library is time-consuming if the policy library is large and the building environment has multiple zones. For example, if an online policy selection algorithm like UCB [3] is used on a building with m thermal zones and a policy library with n policies, then the total number of arms to represent all possible policy assignments would be n^m. For the reasons mentioned above, we restrict our focus to the offline policy selection problem.

Recent work attempted to combine offline policy evaluation OPE with online selection. Konyushova et al. [25] evaluate the policies via OPE first and employ a UCB-like Gaussian Process (GP) to select the policies in an online fashion. However, the UCB algorithm is not suitable for the building control domain due to the reasons mentioned previously. Paine et al. [60] propose a method to estimate a distribution of the value of each policy. However, in the building control domain, we only focus on policies with a high probability of being optimal. Therefore, we believe that a point estimate of the value of each policy would suffice.

4.2 Policy Ranking via Zero-Cost Proxies

In the RL setting, the loss cannot be calculated without running the policy in the target environment. This prevents ZCPs from being directly used for evaluating policies. We overcome this limitation by re-weighting the rewards obtained in the offline dataset using the importance sampling ratio $\rho(s, a)$ defined in Section 2.1.6. Concretely, we sample a trajectory from $\mathcal{D}$ and re-weigh the rewards a follows:

$$\hat{r}_t = \rho(s_t, a_t) r_t, \tag{4.1}$$

where $\hat{r}_t$ is the re-weighted reward at time step t, r_t is the reward obtained at time step t, and $\rho(s_t, a_t)$ is the importance sampling ratio at time step t. By replacing the actual rewards with the re-weighted rewards in the trajectory $\mathcal{D}$, we get a modified trajectory that acts as a proxy for deploying the evaluation policy on the target environment. The modified trajectory is then

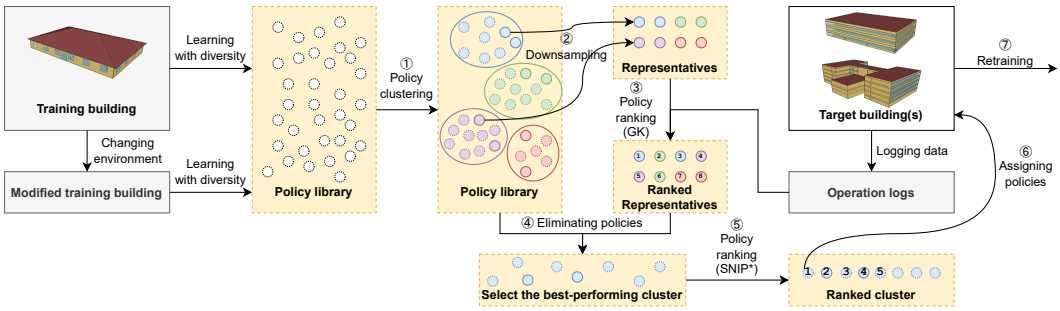


Figure 4.1: Schematic overview of the proposed methodology where circled numbers show different steps of the methodology

used to calculate the PPO loss $\mathcal{L}_{PPO}$ defined in Equation 2.1. This loss is then backpropagated through the policy network to calculate the gradients for each layer, which are used by the gradient-based ZCP methods (GN and SNIP). For clarity, we distinguish GN and SNIP methods that use the proxy trajectory by referring to them as GN* and SNIP*, respectively. The scalar values obtained for each policy in the library are then used to obtain the policy ranks $\mathcal{O}$. To select the best-estimated policy, we select the policy with the highest rank, i.e., $\pi^* = \arg\max_{\pi \in \Pi} \mathcal{O}(\hat{\mu}(\pi))$. Recall that $\hat{\mu}(\pi)$ is the estimated value of the policy π.

To study the performance of using ZCPs for policy evaluation, we compare the performance of GN* and SNIP* with the standard OPE methods, namely IPW and SNIPW on the three building environments introduced in Section 3.2.

4.3 Empirical Validation

In this section, we first outline the overall policy selection methodology under the building control framework explained in Section 3.1. We then compare the performance of the proposed method with the OPE methods.

4.3.1 Methodology

We follow the warm-start policy selection algorithm for MARL-based control of the HVAC system proposed in [16]. The methodology has three main parts: 1) building a diverse policy library, 2) policy selection using offline data, and 3) policy transfer and retraining. In our study, the offline data is two weeks

of operational log data generated using the rule-based controller in the target building. We first build a policy library, then use a clustering algorithm and different policy evaluation algorithms to select the best set of policies for transfer. The policy clustering algorithm is used to reduce the number of policies that need to be evaluated as this process may be repeated every few months or seasons. After these policies are transferred to their respective zones in the target building, they are trained on the target building in an online fashion. Figure 4.1 shows a schematic overview of our methodology.

Building a Diverse Policy Library

Recall that policy diversity-induced RL can train several policies with varying skills. In this thesis, we utilize policy diversity proposed in [66] to generate a library of policies. Similarly, environment diversity is also incorporated when training the policy library. We incorporate environment diversity by installing blinds to cover all windows in the training environment. This technically adds a new training environment. Furthermore, we update the occupancy pattern of each zones to remove the time intervals when a zone becomes unoccupied (e.g., lunchtime) during core business hours. This gives us two more training environments, bringing the number of environments to four.

In our implementation, Building A depicted in Figure 3.2a is used to build the policy library considering both policy and environment diversity. All policies are trained using PPO under the MARL framework for 1,000 episodes to ensure convergence. We consider three policy diversity weights (refer to Section 2.1.5) $w \in 0.1, 1, 10$ to identify diverse policies. These policies are forced to be different from the optimal policy $\pi^*(w = 0)$ that is learned for the given zone. This results in 800 policies in the policy library — 10 random seeds for training $\times$ 4 training environments $\times$ 5 zones per environment $\times$ 4 diversity weights.

Policy Clustering

We employ policy clustering to remove the need for evaluating all the policies. This is necessary as this entire process may be repeated every few months or seasons, and since the policy library can be large, an efficient approach to select the best policy for each zone is required. We cluster policies according to

their behavior in their training environment(s). We represent each policy in the policy library using a feature vector of length m. This vector is constructed by sampling $m-1$ states from the distribution of visited states when the policies were being trained. The last element of the feature vector is the initial state of the target environment. We then use the actions that would be taken from these m states under the policy to construct the feature vector. We set m to 10 in this work. We then use K-Means to cluster all policies in the policy library. The elbow method is used to determine the number of clusters.

Once the clusters are formed, $n-1$ random policies from each cluster are sampled, and the centroid policy are taken as the set of representative policies for each cluster. The centroid policy is selected as it may represent the average performance of the cluster in the training environment, and the other randomly selected policies increase the confidence in the evaluation result. We set n to 5 in this study.

Policy Evaluation and Selection

In our proposed methodology described in Figure 4.1, there are two places where policy evaluation takes place, namely Step 3 and Step 5. In Step 3, we rank the representative policies from each cluster to obtain the ranking of clusters, whereas, in Step 5, we only rank the policies from the top cluster. To rank the policies two weeks of operational log data collected under the rule-based controller in the target building is used. The best-performing policy from the top cluster is then transferred over to the target environment. All the steps shown in Figure 4.1 are repeated for each zone in the target building.

Policy Transfer and Retraining

After assigning the best estimated policies for each zone in the target building, we retrain all policies using the MARL framework in an online fashion. Updating the policies through interaction with the target building allows the policies to further adapt to the new environment, improving their performance.

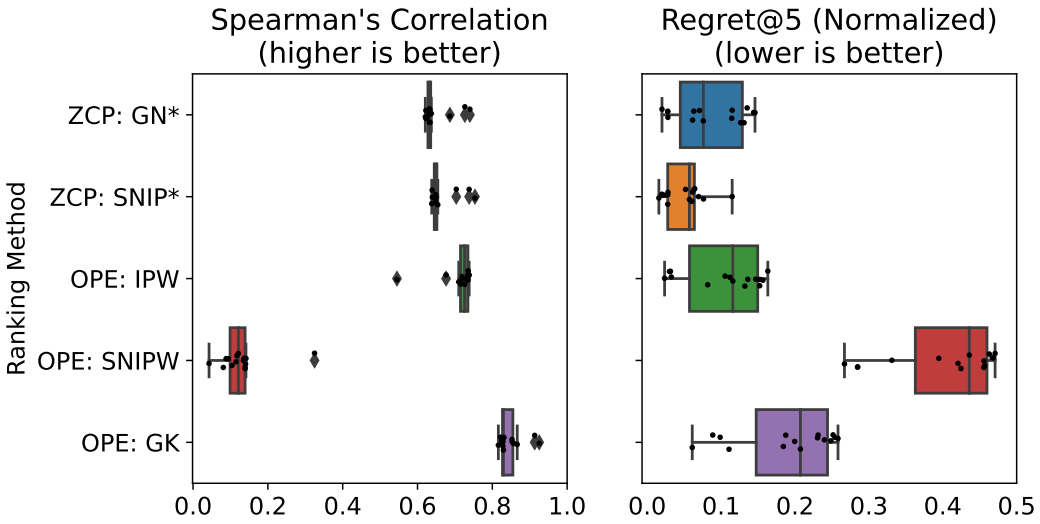


Figure 4.2: The evaluation metrics for policy ranking methods where each dot represents the score for a zone. The ground truth ranking was obtained by manually testing each policy in the policy library on $\mathrm{B}_{\mathrm{Denver}}$.

4.3.2 Comparing Policy Ranking Methods

To test how well the newly proposed method works for policy selection, we compare its performance with the three OPE methods introduced in Section 2.1.6. The first two are IPW and SNIPW that assume the action space is discrete. We discretize the action space using the Freedman Diaconis estimator [12]. The third standard OPE method that we test is Gaussian Kernel (GK), a Gaussian kernel estimator with a bandwidth of 0.3. The GK method can work with continuous action spaces. In this section, we compare the efficiency of the five policy ranking methods, namely IPW, SNIPW, GK, GN*, and SNIP*.

First, we obtain the ground truth ranking of the policies in the policy library by deploying them to the target environment and observing their expected return. This is a brute force approach, which is computationally expensive. Then using the estimated ranking of the policies obtained from the five methods, we calculate the Spearman's correlation and Regret@k, introduced in Section 2.1.7, for each of the above-mentioned methods.

Figure 4.2 (left) shows Spearman's rank correlation for all the ranking methods. Recall, the Spearman's rank correlation measures how well the ranking of the policies obtained from the OPE methods matches the ground truth ranking. The higher the Spearman's rank correlation, the better the ranking

method. Among the OPE methods, IPW has a mean value of 0.71, while SNIPW has a mean of 0.12, indicating that IPW performs considerably better than SNIPW. The GK method has a mean of 0.84 and performs better than IPW with statistical significance ($P < 0.0001$). The two ZCP methods have similar performance with GN* and SNIP* having a mean Spearman's rank correlation of 0.65 and 0.66, respectively. From this figure, we conclude that the GK method performs the best in terms of Spearman's rank correlation.

Figure 4.2 (right) compares the Regret@5 metric for all the ranking methods. Recall, that the Regret@5 metric measures how well the ranking method can identify the best-performing policies. The lower the Regret@5, the better the ranking method. The regret values are normalized according to the maximum difference between the actual value of the best and worst policies in the ground truth set. We see that the GK method with a mean value of 0.19 is not the best-performing method. Instead SNIP* yields a mean regret of 0.05, which is the lowest among the ranking methods. When comparing SNIP* with GN*, the former has a lower regret with statistical significance ($P = 0.018$).

Although the GK method has the highest Spearman's rank correlation, SNIP* outperforms the OPE methods when it comes to regret. To take advantage of GK's high Spearman's rank correlation as well as the ability of SNIP* to more accurately identify the best performing policies, we employ GK for Step 3, and SNIP* for Step 5 of our proposed methodology, as shown in Figure 4.1.

4.3.3 Policy Clustering Analysis

We consider building B_{Denver} for this analysis. The elbow method suggests clustering our policy library into six clusters for all zones. After ranking representative policies from each cluster in Step 3, we only consider the best-performing cluster. Given that the elbow method yielded 6 clusters, Step 4 eliminates 83% of the policies in the policy library.

4.3.4 Policy Transfer Analysis

In the previous section, we concluded that using GK for Step 3 and SNIP* for Step 5 of our proposed methodology produces the best results. We refer to

this approach as GK-SNIP*. We compare the zonal control policies selected from the policy library by GK-SNIP* with four baselines, namely:

Default: The default rule-based controller in the building model

MARL: Independent zone-level control policies that are trained from scratch

SARL: A single policy that controls the entire building trained from scratch

Best GT: Representing the set of best independent zone-level control policies for the target building from the ground truth ranking of the policy library. The ground truth ranking is obtained via brute force evaluation of all policies in the policy library on the target building. This represents the lower bound of the energy consumption that can be achieved by the proposed method.

The last baseline is hypothetical as it gives a lower bound on building energy consumption using the proposed method. If we beat the first baseline, it means that our policies can save more energy than the controller commonly used in practice, without sacrificing thermal comfort. In this analysis, we study the performance of our proposed method only for the month of January for the three target buildings. Exploration of the performance of the proposed method for other months and a full year is done in Chapter 5.

Figure 4.3 shows the performance of our proposed method with other baselines in the target building B_{Denver}, which is different from the source building, Building A as mentioned in Section 3.2. All policies, either initialized randomly or selected from the policy library are (re)trained for 500 episodes (500 months). Note policies that need extensive training are not suitable for deployment on real buildings. For instance, the SARL controller trained on the target building reaches the same level of performance as the Best GT policies assigned from the policy library only after 15,000 episodes, i.e., 1,250 years after deployment!

It can be seen that the proposed policy selection and transfer method provides a reasonable assignment for all zones in B_{Denver}. The performance of the proposed GK-SNIP* method at episode 0 (5.41 MWh) is 22.5% better than the default controller that is presumably designed by HVAC engineers (6.98

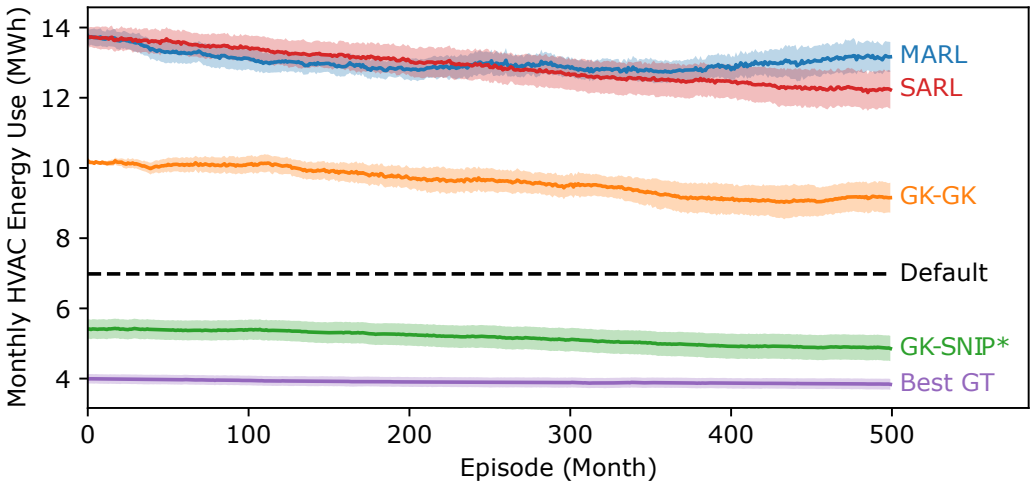


Figure 4.3: Learning curve of different controllers on Building B_{Denver}. Each solid line shows the average performance of 15 runs and the shaded area shows one standard error from the mean. The y-axis is exaggerated.

MWh). It is also significantly better than SARL (13.74 MWh) and MARL (13.77 MWh). This suggests that GK-SNIP* can be employed to select policies that have reasonable performance on the target building. The Best GT assignment has an initial total energy cost of 3.99 MWh. The difference between the proposed policy selection method and the Best GT selection is in part due to the way we sample the policies from the top cluster. Note that the policies assigned to the target building under the Best GT assignment do not benefit significantly from retraining. Specifically, the total HVAC energy consumption reduces by 3.8% (from 3.99 MWh to 3.84 MWh) after 500 episodes. We believe that this is because there is not much room for improvement as we are already close to the minimum HVAC energy consumption that could be realized by a controller in this building given its occupancy schedule and comfort requirements.

The performance of zonal control policies selected by GK-SNIP* improves by 10.2%, reaching the total monthly energy consumption of 4.86 MWh after 500 episodes of training on Building B_{Denver}. This is 30.4% less than the energy consumption of the default controller. Policies trained only on B_{Denver} (not transferred from Building A) fail to reach a level of performance that is comparable with the default controller at the end of the 500 episodes. Specifically, SARL reaches 12.23 MWh and MARL reaches 13.17 MWh. We also witness an increase in energy consumption under MARL after around 200 episodes.

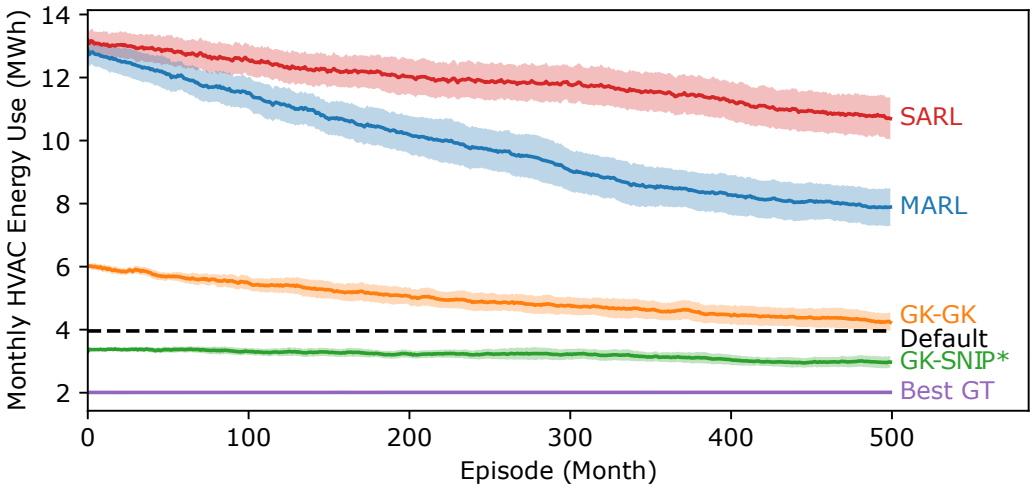


Figure 4.4: Learning curve of different controllers on Building $B_{SanFrancisco}$. Each solid line shows the average performance of 15 runs and the shaded area shows one standard error from the mean.

This might be because agents are not collaborating. As a result, they start to cancel out each other's actions (creating a "fighting zones" situation), thereby increasing the total HVAC energy usage.

To further validate our proposed methodology, we consider two target buildings ($B_{SanFrancisco}$ and C) that have some major differences from the source building (Building A). Building $B_{SanFrancisco}$ is located in a warmer climate than the source building. Moreover, it differs from the source building in terms of the floor area and HVAC design. Building C is a real building where some of the differences include size, occupancy, floor plan, HVAC design, and weather conditions.

Figure 4.4 depicts the performance comparison in Building $B_{SanFrancisco}$. The total energy consumption in all cases is lower than Figure 4.3 because we are looking at a winter month with a higher average outside temperature in San Francisco, reducing the heating demand of the building. Most of the observations made in Section 7.4 are true in this case. Before retraining, the zonal control policies selected from the policy library by GK-SNIP* achieve 16.4% lower monthly energy consumption (3.31 MWh) than the default controller (3.96 MWh). The Best GT assignment yields the lowest monthly energy consumption at episode 0 (2.01 MWh), which is 49.2% lower than the default controller. After 500 episodes of training, the policies selected by GK-SNIP* reduce the total HVAC energy consumption by 10.3%, reaching 2.97 MWh.

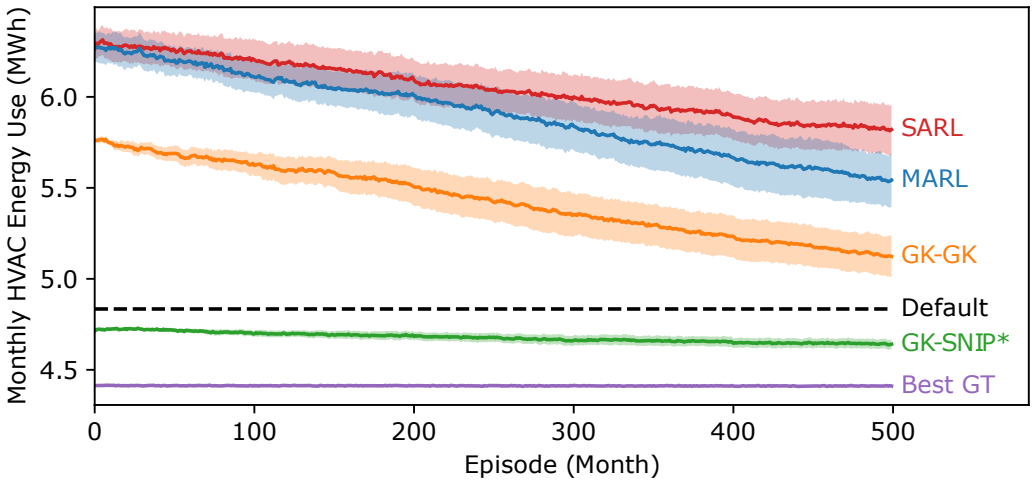


Figure 4.5: Learning curve of different controllers on Building C. Each solid line shows the average performance of 15 runs and the shaded area shows one standard error from the mean. The y-axis is exaggerated.

This is 25.0% lower than the energy consumption of the default controller, yet 50.8% higher than the Best GT assignment.

Figure 4.5 compares the performance of the proposed method with the four baselines in Building C. The same observation can be made here too. GK-SNIP* performs better than the default controller, SARL, and MARL, and is slightly worse than the Best GT assignment. The default controller consumes 4.83 MWh of energy in one month, whereas the proposed GK-SNIP* method reduces it to 4.71 MWh before retraining and to 4.64 MWh after 500 episodes of training in the target building. These numbers are 4.413 MWh and 4.411 MWh for the Best GT assignment.

Our experiments support the claim that diversity-induced RL offers clear benefits for transferring policies to a novel target building, and that the proposed GK-SNIP* policy selection and transfer method can efficiently identify policies, among the policies in the policy library, that perform relatively well in the novel target building using only 2 weeks of historical data. The transferred policies consistently outperform the default controller in terms of HVAC energy use without sacrificing thermal comfort. This is the case even before these policies are retrained to adapt to the new environment.

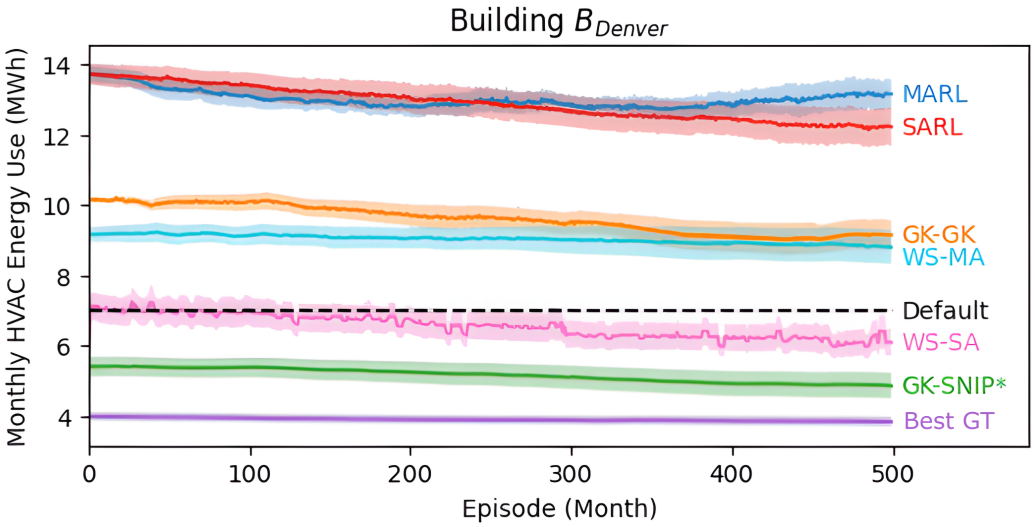


Figure 4.6: Learning curves of different controllers including warm-started policies on Building B_{Denver}. Each solid line shows the average performance of 15 runs and the shaded area. The y-axis is exaggerated.

4.3.5 Comparison to Warm-Starting Policies

To better understand the benefits of the proposed method, we compare it with policies that are warm-started using the offline data on the three target buildings. In this approach, behavior cloning [52], a type of imitation learning, is used to train policies on two weeks of log data collected under the default RBC controller. This is the same amount of data that is used for policy evaluation in our approach, making it a fair comparison. We warm-start the SARL and MARL policies, and call them WS-SA and WS-MA, respectively.

We found that the performance of warm-started policies vary greatly across the buildings. For example, in Figure 4.6, WS-MA performs slightly better than GK-GK and significantly worse than the default controller and GK-SNIP* (our approach) in building B_{Denver} and building C shown in Figure 4.8. However, in building $B_{SanFrancisco}$ shown in Figure 4.7, WS-MA performs 5.1% (3.76 MWh) and 22.0% (3.09 MWh) better than the default controller at episode 0 and episode 500 respectively. For WS-SA, its performance is close to the default controller's performance and improves only slightly over the 500 episodes of retraining in Building B_{Denver} and Building $B_{SanFrancisco}$. In Building C, WS-SA and WS-MA perform similarly. Nonetheless, in all three buildings, our approach (GK-SNIP*) exhibits the best performance across the

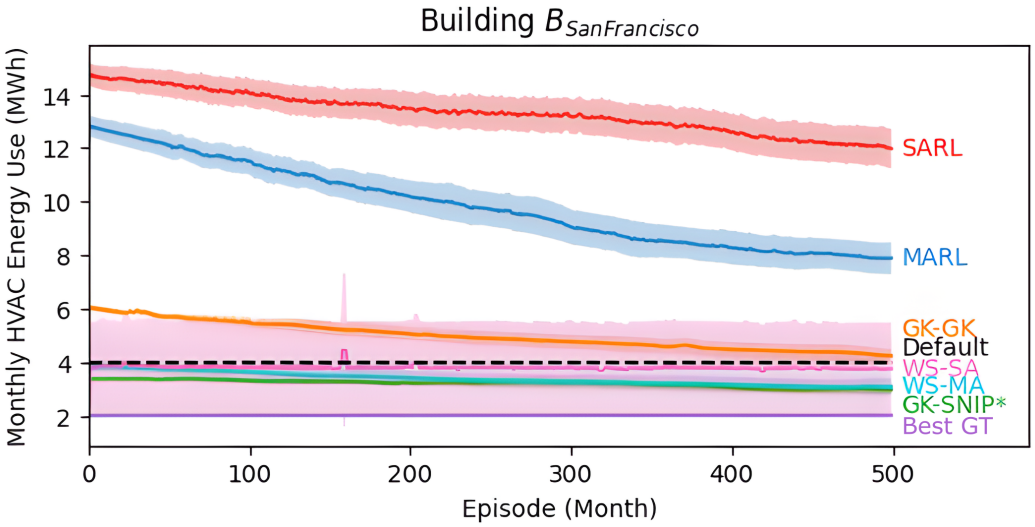


Figure 4.7: Learning curves of different controllers including warm-started policies on Building $B_{SanFrancisco}$. Each solid line shows the average performance of 15 runs and the shaded area. The y-axis is exaggerated.

500 episodes that we considered.

We have also investigated the performance gap at episode 0 between the default controller which generated the log data and the warm-started policies, and concluded that it is most likely due to the small amount of log data used for behavior cloning. Specifically, we increased the amount of data from 2 weeks to 1 month on Building B_{Denver} and Building $B_{SanFrancisco}$ and observed a 9.6% improvement in the performance of the warm-started policies at episode 0. This is interesting as it highlights the fact that our methodology can realize more savings using less log data than RL policies that are trained in an online fashion. That said, further evaluation is warranted to fully understand the sensitivity of the results to (a) the quality of the rule-based controller that generated the log data, and (b) the amount of log data used for behavior cloning. Thus, we defer the full analysis of different imitation learning approaches, the quality of log data, and different kinds of expert demonstration to future work.

4.3.6 Discussion

We now study the effect of using different policy ranking methods in the OPE evaluation steps. In particular, we look at the result of using GK in both steps (labeled GK-GK) and using SNIP* in both steps labeled (SNIP*-SNIP*).

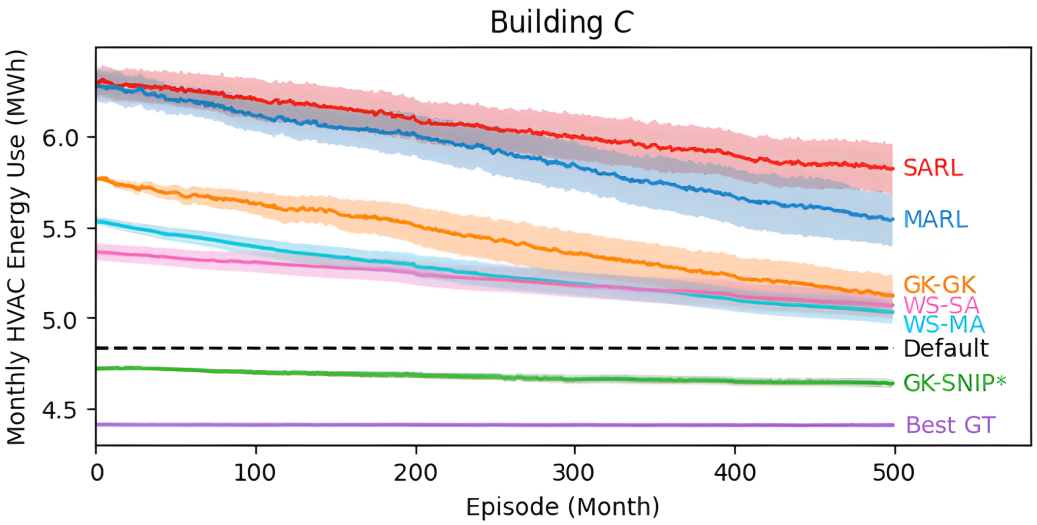


Figure 4.8: Learning curves of different controllers including warm-started policies on Building C. Each solid line shows the average performance of 15 runs and the shaded area. The y-axis is exaggerated.

From Figures 4.3, 4.4, and 4.5, it can be readily seen that GK-SNIP* always achieves lower monthly energy consumption than GK-GK. This is true before and after retraining on the target environment. We attribute this to the fact that once the best-performing cluster is identified, the Regret@5 metric becomes more relevant as we aim to identify the best-performing policies.

It is important to point out that the difference between GK-SNIP* and SNIP*-SNIP* is not significant. For this reason, we remove SNIP*-SNIP* from all figures so that the GK-SNIP* results can be better seen. The difference between GK-SNIP* and SNIP*-SNIP* becomes more pronounced as the number of episodes for training increases. But even after 500 episodes of training, the policies selected by SNIP*-SNIP* consume only 1.0% (4.91 MWh), 3.4% (3.07 MWh), and 0.2% (4.65 MWh) more energy in building B_{Denver}, $B_{\text{SanFrancisco}}$, and C respectively than the policies selected by GK-SNIP*. We attribute this small gap to the fact that the top cluster is well-separated from the other clusters; therefore, the first round of evaluation is more robust to potential estimation errors.

Chapter 5

Analyzing the Sensitivity to Seasonality

In the previous chapter, we presented a method for selecting a near-optimal policy to control the HVAC system of a specific target building. While we demonstrated the performance of our selection algorithm after deploying it on several buildings, they were only deployed for a single month — January. In this chapter, we address the challenge of deploying selected policies for a whole year, where we do not have control over the month from which the log data were collected. Specifically, we investigate the importance of log data in policy selection performance, present a few solutions to address the challenges that arise due to changing log data, and conclude by comparing their performance.

5.1 Impact of Log Data

To obtain the results presented in Section 4.3, we used 15 days of log data collected in January from the target building for policy ranking and policy selection. In this section, we study how the performance of policy selection changes when the month from which the log data is obtained changes. The primary metric used for comparison is the Regret@k metric introduced in Section 2.1.7. The Spearman's rank correlation is no longer necessary because our problem now is identifying the best-performing policy, not the overall ranking of the policy library, which would have been helpful if we were searching for the best cluster.

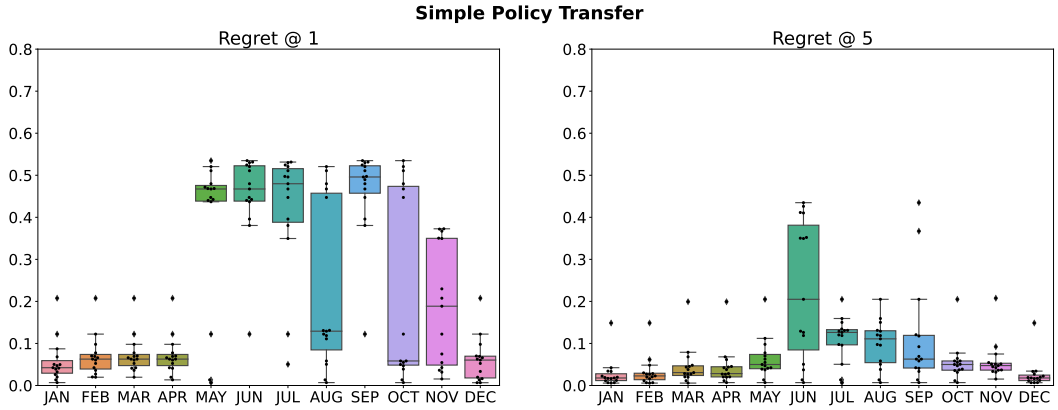


Figure 5.1: The regret values for standard policy selection algorithm. The x-axis shows the month in which the log data is collected. The ground truth ranking was obtained by manually testing each policy in the policy library on B_{Denver} for a full year.

The ground truth dataset was obtained by manually evaluating each policy in the policy library for an entire year. This dataset is used for calculating the Regret@k value for our policy selection method. Recall that during the brute force evaluation of all the policies in the policy library, we assign the policy to one zone and assume a rule-based controller controls all the other zones. This is done to reduce the size of the brute force search space, and we believe that it provides a valid proxy for the actual performance of the policy if it were deployed. The regret values only show how close a method is to the ground truth and cannot be used to draw concrete conclusions about the actual energy consumption. To accurately determine the annual energy consumption, it is necessary to deploy the chosen policies in the target building for the whole year.

Figure 5.1 (left) depicts the Regret@1 values for our policy ranking method across all months. Figure 5.1 (right) depicts the Regret@5 values. These plots show the regret values for the top-1 and top-5 policy selection cases, respectively. Recall that we choose the policy with the highest estimated value in the top-1 policy selection case and deploy it in the building. In the top-5 policy selection case, we obtain five policies with the highest estimated value and choose the best one from this set. This is only possible if we have access to an oracle that can accurately identify the best policy in the top-5 set. Increasing the value of k increases the chance that our proposed policy selection algorithm chooses a better policy. Although such an oracle does not

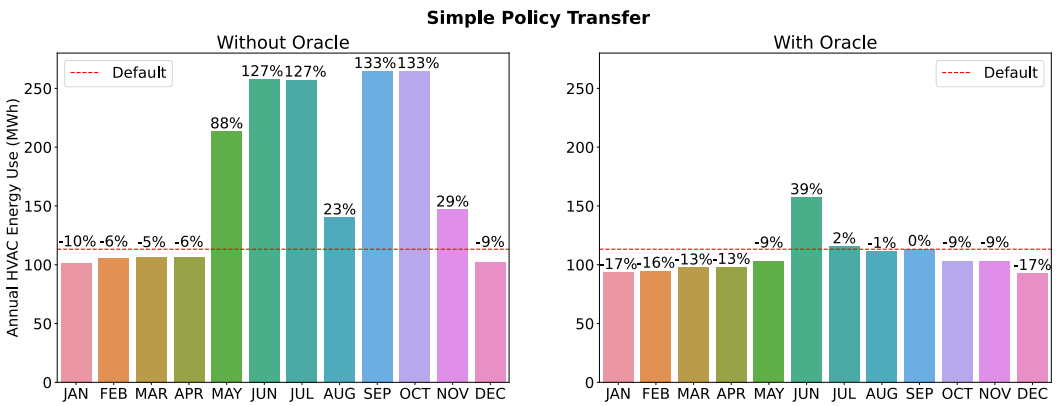


Figure 5.2: The annual energy consumption for our proposed policy selection algorithm in the B_{Denver} environment. The x-axis shows the month from which the log data is collected. The percentages represent the percentage increase or decrease in energy consumption compared to the default rule-based controller, denoted by the red dashed line.

exist in reality, we assume that we have access to such an oracle to better study the impact of log data on policy selection performance. Unless stated otherwise, we use the ground truth dataset as the oracle to identify the best policy in the top-5 set. Approaches such as policy ensembles and online policy selection which can be used for approximating this oracle, are discussed in Section 5.3.

From Figure 5.1 (left), we see that the regret values are low when log data is chosen from the winter and spring months, i.e., December, January, February, March, and May. However, the regret values are much higher during the summer months, i.e., June, July, and August. This tells us that our policy selection algorithm would perform well if the log data were obtained from the winter and spring months. However, if the log data were obtained from the summer and fall months, our policy selection algorithm would perform poorly. From Figure 5.1 (right), the regret values are much lower than the top-1 selection case. However, regret values are still high during the summer months. This indicates that our selection algorithm will still perform poorly during the summer months even if we have access to an oracle that can accurately identify the best policy in the top-5 set.

Figure 5.2 depicts the annual energy consumption for our policy selection method across all months. For this experiment, we calculate the annual energy consumption in building B_{Denver} for when we have an oracle and when we do

not have an oracle. The horizontal red line shows the energy consumption of the default rule-based controller in the building. When comparing annual energy consumption with the corresponding regret values in Figure 5.1, we see that the regret values are a good indicator of actual energy consumption. In Figure 5.2 (left), we see that if we directly choose the best-estimated policy, the energy consumption is much higher than the default controller during the summer and early fall months. Often they consume more than twice the energy consumed by the default controller. If the log data were from these months, our policy selection algorithm would yield policies that consume, on average, 95% higher energy than the default controller.

When we have access to an oracle, depicted in Figure 5.2 (right), we see that the energy consumption for June and July is higher than that of the default rule-based controller. In the other summer months, the decrease in energy consumption is negligible, if any. When we do not have access to an oracle, the average energy consumption across the months is 172.27 MWh with a standard deviation of 72.64. When we access an oracle, the average energy consumption and standard deviation are 107 MWh and 17.63, respectively. In comparison, the default rule-based controller has an average energy consumption of 113.21 MWh. Using an oracle yields lower energy consumption with a smaller variance, but in some summer months, the energy consumption is higher than the default controller.

These plots show that our policy selection algorithm would perform well if the log data were obtained from the winter months. This observation is further supported by the decrease in energy consumption observed in the three target buildings tested in Chapter 4. However, if the log data were obtained from the summer months, our policy selection algorithm would perform poorly. It is worth nothing that these results are building-specific and this conclusion may not be valid for other buildings or weather patterns. The conclusion we draw from these results is that our policy selection algorithm is highly dependent on the month from which the log data is obtained. Our proposed approach is not robust to changes in the log data, and this is one of its limitations. The following sections will explore solutions to reduce this variance and compare their performance.

5.2 Reducing the Variance in Performance

In this section, we focus on reducing the variance in policy selection performance when the month from which the log data is obtained changes. First, we introduce the idea of swapping policies, where instead of deploying the selected policies for an entire year, we deploy policies for a month and then re-evaluate with the new log data and choose new policies for the following month. We then introduce another approach where instead of using the heating energy consumption as the reward, we use a different reward signal that incorporates cooling energy consumption, as discussed in Section 3.1. We then discuss other approaches that can be used and conclude by comparing these approaches regarding annual energy consumption.

5.2.1 Policy Rotation

Policy rotation is a straightforward approach where, instead of committing to a single policy for the entire year, the policy is changed every month. The offline log data is only used initially to select the first set of policies for each zone. After that, the log data generated by the policies currently deployed in the building is used for selecting policies for the following month. This method is advantageous because all the log data generated by the policies currently deployed can be used for re-evaluation and selection. We conjecture that because weather patterns of consecutive months are correlated, policy rotation might be beneficial for reducing the variance in policy selection performance.

Algorithm 1: Policy rotation

inputs : Π: Diverse policy library;
π^{init}: The initial policy assignment obtained via OPE;
env: target building environment wrapper;

$\pi^{\text{current}} \leftarrow \pi^{\text{init}}$ `// initialize the current policy assignment`
for *month* $\in 1 \dots 12$ **do**
  $S_{t..t+n}, A_{t..t+n}, R_{t..t+n}, S'_{t..t+n} \leftarrow \text{env.evaluate}(\pi^{\text{current}})$
  $\pi^{\text{current}} \leftarrow \text{policy\_selection}(S_{t..t+n}, A_{t..t+n}, R_{t..t+n}, S'_{t..t+n}, \pi^{current}, \Pi)$
end

Algorithm 1 is the policy rotation algorithm. Using Building B_{Denver} as the test environment, we calculated the Regret@1 and Regret@5 values for this algorithm, depicted in Figure 5.3. To obtain each month's ground truth

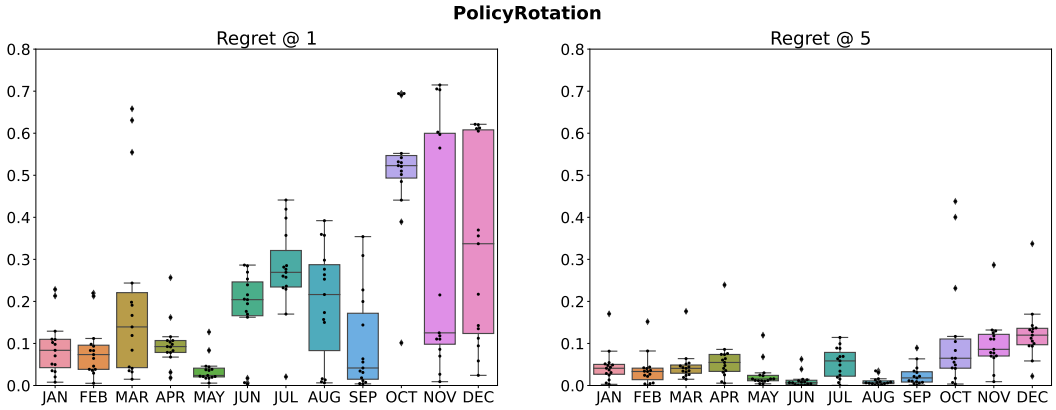


Figure 5.3: The regret values for our policy ranking given log data across all months in the B_{Denver} environment when using policy rotation. The x-axis shows the month from which the log data is collected. The regret values in this case are calculated for each month separately.

policy ranks, we ran a brute force search for each month independently. In other words, we evaluated each policy in the policy library for each month separately and then ranked them according to their energy consumption. From Figure 5.3 (left), we see that if we directly choose the policy with the highest estimated value each month, the regret values vary significantly and do not help reduce the performance variance. It is worth noting that, compared to the simple policy transfer method, depicted in Figure 5.1 (left), the regret values are lower during the summer months. Yet, when an oracle is available, depicted in Figure 5.3 (right), we can see that the regret values are much lower across all months. This indicates that policy rotation can reduce the variance in policy selection performance when the log data changes. However, when an oracle is unavailable, policy rotation does not reduce the variance in performance.

5.2.2 New Reward Signal for OPE

To uncover why our offline policy selection algorithm performs poorly in the summer months, we inspected the state, action, and reward of the agent in each zone of the building over the course of a year. It was found that during the summer months, the rewards received by the agents were sparse. In other words, the agents did not receive a value other than zero for most of the time steps during the summer months. Recall that the reward signal used for the

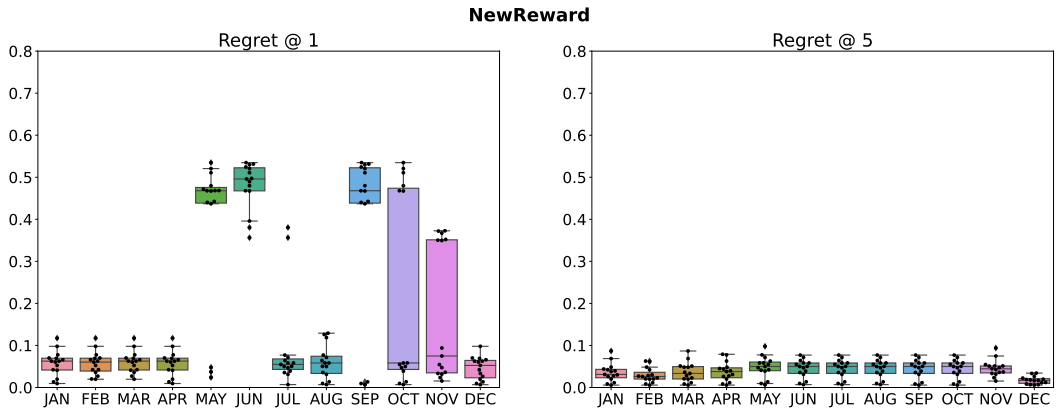


Figure 5.4: The regret values for our policy ranking given log data across all months in the B_{Denver} environment using the new reward signal. The x-axis shows the month from which the log data is collected. The ground truth ranking was obtained by manually testing each policy in the policy library on B_{Denver} for a full year.

agents in the building is the heating energy consumption, as mentioned in Section 3.1. Because some zones do not require heating, the log data produced by the rule-based controller does not contain enough information for our proposed offline policy selection algorithm.

Recall that in Section 3.1, we defined a new reward signal that incorporates both heating and cooling energy consumption, shown by Equation (3.2). To alleviate the problem of sparse reward signals during the summer months, we use this new reward signal for the OPE phase of the policy selection algorithm. A limitation of this new reward is that in buildings where the AHU system supplies fresh air to all zones, the cooling energy consumption will be the same for all the zones. In other words, the new reward signal might be the same for some of the zones in the building. Another limitation is that since the agents have not been trained with the new reward signal, performing offline policy evaluation with the new reward signal might not be accurate. However, from empirical tests depicted in Figure 5.4, we found that the new reward signal benefits policy selection.

Figure 5.4 depicts Regret@1 and Regret@5 values for our policy ranking method across all months when using the new reward signal for OPE in building B_{Denver}. The ground truth ranks were obtained by brute force evaluation of each policy in the diverse policy library for an entire year. In other words, we use the new reward signal to choose a policy from the policy library and

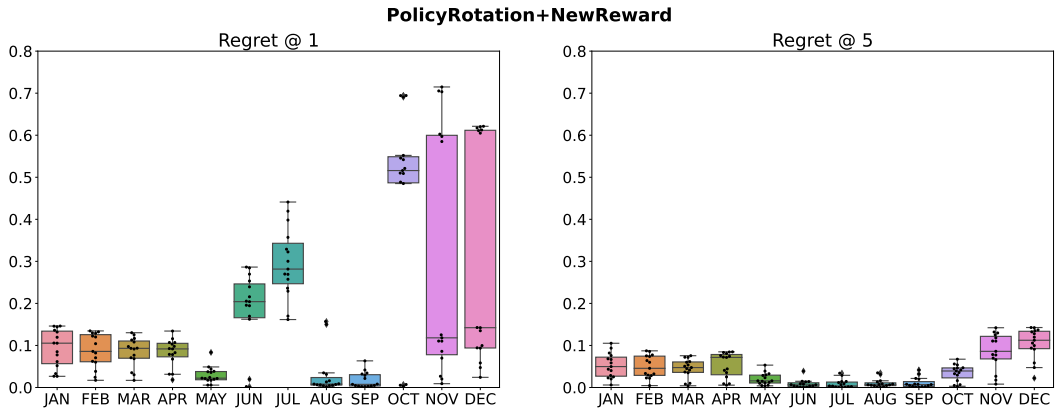


Figure 5.5: The regret values for our policy ranking given log data across all months in the B_{Denver} environment using the new reward signal and policy rotation. The x-axis shows the month from which the log data is collected. The ground truth ranking for each month was obtained by manually testing each policy in the policy library on B_{Denver} for a month.

then calculate the regret based on its full-year energy consumption. In Figure 5.4 (left), we see that even with the new reward signal, the policy selection performance is poor during the summer months when an oracle is unavailable. In some summer months, i.e., July and August, the regret values are much lower than the simple policy transfer method depicted in Figure 5.1. However, when an oracle is present, the performance is significantly better, as presented in Figure 5.4 (right). The regret values are much lower during the summer months, and there is low variance throughout the year. Another noteworthy observation is that far fewer outliers are present when an oracle is available compared to Figure 5.3 (right). This is most likely because the new reward signal alleviates the sparseness of rewards for log data collected in the summer months. These preliminary results indicate that policy selection using this new reward signal with an oracle may be a viable solution for reducing the variance in policy selection performance.

5.2.3 Policy Rotation with New Reward

The following approach we consider combines the two approaches discussed previously. Instead of using the new reward signal for selecting a policy for the entire year, we can use it for the OPE phase of the policy rotation algorithm at the start of every month.

Figure 5.5 (left) and (right) depict Regret@1 and Regret@5 values for our policy ranking method across all months when using the new reward signal in the policy rotation algorithm respectively. We ran a brute force search for each month independently to obtain each month's ground truth policy ranks. The left figure shows that the regret values are much higher during the summer months when an oracle is unavailable. However, compared to the simple policy transfer approach, depicted in Figure 5.1 (left), PolicyRotation+NewReward has much lower regret values in the summer months. When an oracle is available, presented in Figure 5.5 (right), it is clear that policy rotation with the new reward signal is beneficial for reducing the variance in policy selection performance. These results indicate that combining these two approaches provides yet another viable solution for reducing the variance in policy selection performance, provided that an oracle is available to choose the best policy from the top-5 set.

5.2.4 Comparing Annual Energy Consumption

In the previous sections, we have only discussed the regret values for the different approaches. Although regret may be a good indicator of policy selection performance, it does not provide information about actual energy consumption. In this section, we will compare the annual energy consumption of the different approaches discussed in this chapter in Building B_{Denver}.

We use two baselines for comparison, i.e., the default rule-based controller and the policies with the lowest energy consumption for each zone for the entire year. We refer to the default rule-based controller as *Default* and the policies with the lowest energy consumption as *Best GT*. The Best GT is determined by brute force evaluation of all the policies in the policy library for each zone for an entire year. The policies with the lowest energy consumption are chosen as the Best GT policies. Thus, the Best GT policies are the best possible policies obtained from the policy library.

Ideally, the proposed solutions for reducing the variance should lie between the Default and Best GT baselines. We mark our proposed policy selection approach as *SimplePolicyTransfer*. Recall that policies chosen initially using the offline log data are deployed for an entire year in the simple policy transfer approach. We mark the policy rotation approach as *PolicyRotation*,

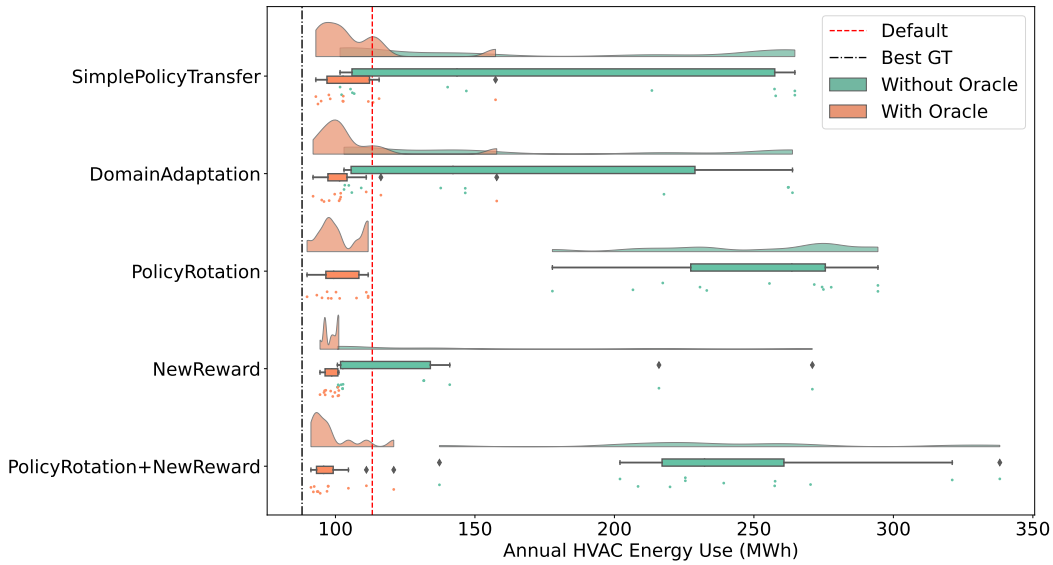


Figure 5.6: The average energy consumption for our policy ranking across all months in the B_{Denver} environment. The percentages represent the percentage increase or decrease in energy consumption compared to the default rule-based controller. The error bars represent one standard deviation of the annual energy consumption. Note the y-axis is exaggerated.

and the new reward approach as *NewReward*. The combination of both approaches is marked as *PolicyRotation+NewReward*. We also introduce another approach, called *DomainAdaptation*, where the policies chosen by the simple policy transfer approach are trained in an online fashion on the new data from the target building. Specifically, the policies are retrained every month on the data generated by deployment for the month prior. For the approaches mentioned above, we repeat the process with log data from each month of the year and calculate the annual energy consumption. This yields 12 different annual energy consumption values for each approach. Figure 5.6 is a box plot depicting annual energy consumption distribution for the different approaches.

For all the proposed solutions discussed previously, we calculate the annual energy consumption for cases where an oracle is available and where an oracle is not available. When an oracle is unavailable, the simple policy transfer approach has an average annual energy consumption of 172.27 MWh with a standard deviation of 72.64, which is around 52.26% higher than the default rule-based controller. When domain adaptation is used on these policies, the average annual energy consumption is 163.62 MWh with a standard deviation of 67.93. Domain adaptation reduces the average energy consumption

by 5.0% compared to the simple policy transfer approach without an oracle. Although domain adaptation reduces energy consumption, the value is still higher than the default rule-based controller, which has an energy consumption of 113.21 MWh. The PolicyRotation approach has an average annual energy consumption of 250.73 MWh with a standard deviation of 37.24. This is around 121.75% higher than the default rule-based controller. The worse performance compared to the simple policy transfer approach without oracle is likely because once bad policies are chosen, they generate low-quality log data, which is then used for policy selection for the following month. The NewReward approach, on the other hand, has an average annual energy consumption of 133.67 MWh with a standard deviation of 54.60. The NewReward approach without oracle shows promise because the annual energy consumption values are close to the rule-based controller. The NewReward approach only has an average 18.04% increase in energy consumption compared to the rule-based controller. The PolicyRotation+NewReward approach has an average annual energy consumption of 241.86 MWh with a standard deviation of 53.53. This is similar to the PolicyRotation approach, and the reason for this is the same as the PolicyRotation approach.

When an oracle is available, the simple policy transfer approach has an average annual energy consumption of 107 MWh with a standard deviation of 17.63, which is around 5.44% lower than the default rule-based controller. Even though the average annual energy consumption is lower than the default rule-based controller, the standard deviation is high, indicating that for log data obtained from certain months, the annual energy consumption will be higher than the default rule-based controller. When domain adaptation is used, the average energy consumption is 106.02 MWh with a standard deviation of 17.62. Domain adaptation shows promise as it reduces energy consumption; however, outliers in the box plot indicate that the energy consumption is much higher for some months than the default controller. The PolicyRotation approach has an average annual energy consumption of 101.25 MWh with a standard deviation of 7.55. This approach is much better than the simple policy transfer approach, as the standard deviation is low, ensuring that this approach will not perform much worse than the default rule-based controller. It is worth noting that PolicyRotation outperforms domain adaptation, indicating that re-selecting new policies leads to lower energy consumption than retraining policies. Another noteworthy observation is that there are no outliers in the

box plot, indicating that the energy consumption will be lower than the default rule-based controller no matter which month the log data is obtained from. The NewReward approach has an average annual energy consumption of 98.39 MWh with a standard deviation of 2.43. This is the best-performing approach, as it yields the lowest average energy consumption and the lowest standard deviation. The PolicyRotation+NewReward approach has an average annual energy consumption of 98.82 MWh with a standard deviation of 9.03. This is slightly better than the PolicyRotation on average. However, some outliers in the plot indicate that the energy consumption is higher for certain months than the default controller. We conclude that the best-performing method on building B_{Denver} is NewReward, which has the lowest average energy consumption and the smallest standard deviation.

When comparing the methods that use an oracle to the ones that do not, we see that using an oracle yields significantly lower energy consumption. This is because we increase the chance of choosing a better policy by increasing the value of k. However, in reality, an oracle that can accurately identify the best policy in the top-k set is not available. In the following section, we will discuss some approaches that can be used to choose the best policy from the top-k set. However, implementing and testing these approaches are left for future work.

5.3 Approximating the Oracle

From Figure 5.6, it is clear that the NewReward approach with an oracle performs the best. However, the oracle must be replaced with a viable alternative for this approach to be used in reality. Recall that Section 4.1 discusses the issues related to online policy selection algorithms in the building HVAC control domain. The main reason UCB, an online selection algorithm, was unsuitable for the original problem is that the number of arms needed to represent the policies for each zone was very large. However, in the top-k policy selection problem, the number of arms is k, significantly smaller than the number of policies in the policy library. Thus, UCB may be a viable solution for replacing the oracle. Konyushova et al. [25] explore using OPE estimates to warm-start the online policy selection process. They show that warm-starting with policy estimates sped up the convergence time of UCB significantly.

Another approach to replace the oracle is policy ensembling. Wiering et al. [57] discuss various ways in which policy ensembles can be used to improve the performance of reinforcement learning algorithms. They show that approaches such as majority voting and rank voting can combine or choose an action among the policies in the ensemble. In another line of work, Zhumabekov et al. [68] use the standard deviation of the actions of policies in the ensemble to weigh the actions of the policies. By adopting these approaches to combine the actions produced by the policies in the top-k set, we may be able to approximate the oracle.

Although these approaches are promising, their implementation and comparison are outside the scope of this thesis and are deferred to future work. In summary, our proposed policy selection algorithm could be made more robust to changes in the log data. However, our proposed policy selection algorithm performs well if the log data is obtained from preferable months. We wish to emphasize that we have studied the problem when little log data is available; our offline policy selection algorithm may be more robust if more log data is available.

Chapter 6

Conclusion

Commercial buildings are responsible for a significant portion of global energy consumption, and the HVAC systems within them are significant contributors to this consumption. Conventional HVAC controllers are rule-based and require expert knowledge to design. They are also less efficient than more advanced controllers, such as MPC and RL. That being said, these advanced controllers require either a building model or a large amount of training data to learn good control strategies. These challenges have made it difficult for the HVAC industry to adopt these controllers. Offline RL methods offer ways to reduce the training time required by RL controllers by using offline log data generated by the existing rule-based controllers. However, a major challenge is that the policies learned by these methods are not transferable to new, unseen building environments.

Policy diversity is vital in allowing policies to generalize to new environments. This thesis addresses the transferability issue with RL-based HVAC controllers by proposing an offline policy selection algorithm that can identify high-quality policies for a new building environment using only a small amount of offline log data and a large library of diverse policies. The novelty of our work is in (a) combining policy clustering and policy evaluation techniques to quickly identify high-quality policies among the diverse policy library for the target building and (b) modifying the standard ZCP-based methods to make them applicable to the RL setting.

We compared the performance of various offline policy evaluation and modified zero-cost proxy methods on a building environment. We found that GK

and SNIP* are the best-performing policy ranking methods. We then ran experiments on three building environments and found that the proposed offline policy selection algorithm effectively identifies high-quality zone-level control policies using only two weeks of log data generated by the rule-based controller in the target building. This resulted in a 4% to 30.4% reduction in energy consumption compared to the rule-based controller, significantly improving energy efficiency. However, the proposed method was only tested with log data from January. Further experiments with log data from other months are required to evaluate the robustness of the proposed method.

To better study the performance of the proposed method for a longer time horizon, we conducted experiments on a single building for a whole year. We found that if the offline log data was obtained from preferable months, e.g., January, the proposed method could efficiently identify high-quality policies for each zone. However, if the month in which the log data was obtained changes, the performance of the proposed method would vary greatly. We explored several solutions to address this variance and propose two promising solutions. First, re-evaluating and swapping policies every month reduced the variance to a great extent. This method made our methodology more robust to the month the log data was obtained. Second, using heating and cooling energy consumption values as the reward signal during offline policy evaluation significantly reduced the variance. Regardless, an oracle to select the best policy among the top-k policies with the best-estimated values is necessary for these methods to work. The approaches mentioned above failed to make our algorithm more robust to changes in log data without an oracle. Some approaches to approximate this oracle were briefly discussed, and their comparison was left for future work.

6.1 Limitations and Future Work

In this thesis, a proof-of-concept solution for transfer learning in the building control domain was proposed and thoroughly evaluated. However, there are certain limitations, such as the lack of robustness to changing log data, the lack of a real-world deployment, and the assumptions made about the building environment, to name a few. Moreover, finding a suitable replacement for the oracle and trying out newer ZCPs for OPE are some areas for future work.

Some limitations of our work are related to our assumptions about the building environment. The first assumption is that all buildings have the sensors necessary to construct the state space. In some cases, buildings may not have all the necessary sensors to implement our approach effectively. Another assumption about the building environment is that the VAV boxes in each zone have the same set of control points. If the control points varied, the actions chosen by policies would have to be changed or mapped to the new set of points.

A significant limitation of our work is the necessity of using an oracle to make our approach more robust to changes in the log data. Such an oracle does not exist in reality, and approximations of the oracle must be made to make our approach more reliable. Section 5.3 discussed various ways to approximate the oracle. However, implementation and comparison of these techniques are deferred to future work.

Some limitations may also lie in how the diverse policy library was constructed. In our work, 800 policies were generated from a single building environment using a simulator. Exploring how the offline policy selection performance changes when the policy library is generated from various buildings is an avenue for future work. Similarly, all the policies in the policy library share the same neural network architecture. Investigating how ZCP-based OPE policy selection performance changes when policies with varying architectures are present is another direction for future work.

Finally, we propose using ZCPs as the OPE method. However, both NAS and OPE are active research areas, and in recent years novel methods have been developed that may work better than the ones used in this thesis. Exploring the other OPE methods and comparing their performance with the ZCP-based methods is an interesting avenue for future work. Similarly, investigating why the ZCP-based methods work better than the standard OPE methods used in this thesis for policy evaluation is another important problem that is left for future work.

References

[1] Mohamed S Abdelfattah, Abhinav Mehrotra, Łukasz Dudziak, and Nicholas Donald Lane. Zero-cost proxies for lightweight NAS. In *International Conference on Learning Representations*, 2021.

[2] Guideline ASHRAE. 36: High performance sequences of operation for hvac systems. *American Society of Heating, Refrigerating and Air-Conditioning Engineers, Atlanta*, 2018.

[3] Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47:235–256, 2002.

[4] Bingqing Chen, Zicheng Cai, and Mario Bergés. Gnu-rl: A precocial reinforcement learning solution for building hvac control using a differentiable mpc policy. In *Proceedings of the 6th ACM international conference on systems for energy-efficient buildings, cities, and transportation*, pages 316–325, 2019.

[5] Bingqing Chen, Priya L Donti, Kyri Baker, J Zico Kolter, and Mario Bergés. Enforcing policy feasibility constraints through differentiable projection for energy optimization. In *Proceedings of the Twelfth ACM International Conference on Future Energy Systems*, pages 199–210, 2021.

[6] 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.

[7] Sourav Dey, Thibault Marzullo, Xiangyu Zhang, and Gregor Henze. Reinforcement learning building control approach harnessing imitation learning. *Energy and AI*, 14:100255, October 2023.

[8] U Doe. Chapter 5: Increasing efficiency of building systems and technologies. *Quadrennial Technology Review: An Assessment of Energy Technologies and Research Opportunities*, pages 143–181, 2015.

[9] Ján Drgoňa, Javier Arroyo, Iago Cupeiro Figueroa, David Blum, Krzysztof Arendt, Donghun Kim, Enric Perarnau Ollé, Juraj Oravec, Michael Wetter, Draguna L Vrabie, et al. All you need to know about model predictive control for buildings. *Annual Reviews in Control*, 50:190–232, 2020.

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

[11] Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. Diversity is all you need: Learning skills without a reward function. *arXiv preprint arXiv:1802.06070*, 2018.

[12] David Freedman and Persi Diaconis. On the histogram as a density estimator: L 2 theory. *Zeitschrift für Wahrscheinlichkeitstheorie und verwandte Gebiete*, 57(4):453–476, 1981.

[13] Qiming Fu, Xiyao Chen, Shuai Ma, Nengwei Fang, Bin Xing, and Jianping Chen. Optimal control method of hvac based on multi-agent deep reinforcement learning. *Energy and Buildings*, 270:112284, 2022.

[14] Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In *International Conference on Machine Learning*, pages 2052–2062. PMLR, 2019.

[15] Guang Geng and GM Geary. On performance and tuning of pid controllers in hvac systems. In *Proceedings of IEEE international conference on control and applications*, pages 819–824. IEEE, 1993.

[16] Aakash Krishna GS, Tianyu Zhang, Omid Ardakanian, and Matthew E Taylor. Mitigating an adoption barrier of reinforcement learning-based control strategies in buildings. *Energy and Buildings*, 285:112878, 2023.

[17] Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*, 2023.

[18] Frank Hutter, Lars Kotthoff, and Joaquin Vanschoren. *Automated machine learning: methods, systems, challenges*. Springer Nature, 2019.

[19] International Energy Agency. Buildings: A source of enormous untapped efficiency potential. https://www.iea.org/topics/buildings, 2022.

[20] Max Jaderberg, Wojciech M Czarnecki, Iain Dunning, Luke Marris, Guy Lever, Antonio Garcia Castaneda, Charles Beattie, Neil C Rabinowitz, Ari S Morcos, Avraham Ruderman, et al. Human-level performance in 3d multiplayer games with population-based reinforcement learning. *Science*, 364(6443):859–865, 2019.

[21] Zhanhong Jiang, Michael J Risbeck, Vish Ramamurti, Sugumar Murugesan, Jaume Amores, Chenlu Zhang, Young M Lee, and Kirk H Drees. Building hvac control with reinforcement learning for reduction of energy cost and demand charge. *Energy and Buildings*, 239:110833, 2021.

[22] Nathan Kallus and Masatoshi Uehara. Intrinsically efficient, stable, and bounded off-policy evaluation for reinforcement learning. *Advances in Neural Information Processing Systems*, 32, 2019.

[23] Nathan Kallus and Angela Zhou. Policy evaluation and optimization with continuous treatments. In *International conference on artificial intelligence and statistics*, pages 1243–1251. PMLR, 2018.

[24] Fazel Khayatian, Zoltán Nagy, and Andrew Bollinger. Using generative adversarial networks to evaluate robustness of reinforcement learning agents against uncertainties. *Energy and Buildings*, 251:111334, 2021.

[25] Ksenia Konyushova, Yutian Chen, Thomas Paine, Caglar Gulcehre, Cosmin Paduraru, Daniel J Mankowitz, Misha Denil, and Nando de Freitas. Active offline policy selection. *Advances in Neural Information Processing Systems*, 34:24631–24644, 2021.

[26] 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.

[27] Kuldeep Kurte, Jeffrey Munk, Olivera Kotevska, Kadir Amasyali, Robert Smith, Evan McKee, Yan Du, Borui Cui, Teja Kuruganti, and Helia Zandi. Evaluating the adaptability of reinforcement learning based hvac control for residential houses. *Sustainability*, 12(18):7727, 2020.

[28] Hoang Le, Cameron Voloshin, and Yisong Yue. Batch policy learning under constraints. In *International Conference on Machine Learning*, pages 3703–3712. PMLR, 2019.

[29] Namhoon Lee, Thalaiyasingam Ajanthan, and Philip Torr. SNIP: Single-shot network pruning based on connection sensitivity. In *International Conference on Learning Representations*, 2019.

[30] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1):1334–1373, 2016.

[31] Sergey Levine, Peter Pastor, Alex Krizhevsky, Julian Ibarz, and Deirdre Quillen. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *The International Journal of Robotics Research*, 37(4-5):421–436, 2018.

[32] Chenghao Li, Tonghan Wang, Chengjie Wu, Qianchuan Zhao, Jun Yang, and Chongjie Zhang. Celebrating diversity in shared multi-agent reinforcement learning. *Advances in Neural Information Processing Systems*, 34:3991–4002, 2021.

[33] Yuxi Li. Deep reinforcement learning: An overview. *arXiv preprint arXiv:1701.07274*, 2017.

[34] Muhammad A. Masood and Finale Doshi-Velez. Diversity-inducing policy gradient: Using maximum mean discrepancy to find a set of diverse policies. *CoRR*, abs/1906.00088, 2019.

[35] Kevin R McKee, Joel Z Leibo, Charlie Beattie, and Richard Everett. Quantifying environment and population diversity in multi-agent reinforcement learning. *arXiv preprint arXiv:2102.08370*, 2021.

[36] 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 *International Conference on Machine Learning*, pages 1928–1937. PMLR, 2016.

[37] 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.

[38] Srinarayana Nagarathinam, Vishnu Menon, Arunchandar Vasan, and Anand Sivasubramaniam. Marco-multi-agent reinforcement learning based control of building hvac systems. In *Proceedings of the eleventh ACM international conference on future energy systems*, pages 57–67, 2020.

[39] Avisek Naug, Marcos Quinones-Grueiro, and Gautam Biswas. A relearning approach to reinforcement learning for control of smart buildings. *Annual Conference of the PHM Society,*, 12(14), 2020.

[40] Kingsley Nweye, Bo Liu, Peter Stone, and Zoltan Nagy. Real-world challenges for multi-agent reinforcement learning in grid-interactive buildings. *Energy and AI*, 10:100202, 2022.

[41] Tom Le Paine, Cosmin Paduraru, Andrea Michi, Caglar Gulcehre, Konrad Zolna, Alexander Novikov, Ziyu Wang, and Nando de Freitas. Hyperparameter selection for offline reinforcement learning. *arXiv preprint arXiv:2007.09055*, 2020.

[42] Giuseppe Pinto, Zhe Wang, Abhishek Roy, Tianzhen Hong, and Alfonso Capozzoli. Transfer learning for smart buildings: A critical review of algorithms, applications, and future perspectives. *Advances in Applied Energy*, 5:100084, 2022.

[43] Doina Precup. Eligibility traces for off-policy policy evaluation. *Computer Science Department Faculty Publication Series*, page 80, 2000.

[44] Samuel Privara, Jan Širokỳ, Lukáš Ferkl, and Jiří Cigler. Model predictive control of a building heating system: The first experience. *Energy and Buildings*, 43(2-3):564–572, 2011.

[45] M.M. Rahman, M.G. Rasul, and M.M.K. Khan. Energy conservation measures in an institutional building in sub-tropical climate in australia. *Applied Energy*, 87(10):2994–3004, 2010.

[46] Jorren Schepers, Reinout Eyckerman, Furkan Elmaz, Wim Casteels, Steven Latré, and Peter Hellinckx. Autonomous building control using offline reinforcement learning. In *Advances on P2P, Parallel, Grid, Cloud and Internet Computing: Proceedings of the 16th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC-2021)*, pages 246–255. Springer, 2022.

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

[48] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

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

[50] Adith Swaminathan and Thorsten Joachims. The self-normalized estimator for counterfactual learning. *Advances in Neural Information Processing Systems*, 28, 2015.

[51] Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, et al. Deepmind control suite. *arXiv preprint arXiv:1801.00690*, 2018.

[52] Faraz Torabi, Garrett Warnell, and Peter Stone. Behavioral cloning from observation. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, IJCAI'18, page 4950–4957. AAAI Press, 2018.

[53] Christina Turley, Margarite Jacoby, Gregory Pavlak, and Gregor Henze. Development and evaluation of occupancy-aware hvac control for residential building energy efficiency and occupant comfort. *Energies*, 13(20):5396, 2020.

[54] Jose R Vazquez-Canteli, Gregor Henze, and Zoltan Nagy. Marlisa: Multi-agent reinforcement learning with iterative sequential action selection for

load shaping of grid-interactive connected buildings. In *Proceedings of the 7th ACM international conference on systems for energy-efficient buildings, cities, and transportation*, pages 170–179, 2020.

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

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

[57] Marco A Wiering and Hado Van Hasselt. Ensemble algorithms in reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 38(4):930–936, 2008.

[58] Daniel A Winkler, Ashish Yadav, Claudia Chitu, and Alberto E Cerpa. Office: Optimization framework for improved comfort & efficiency. In *2020 19th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, pages 265–276. IEEE, 2020.

[59] Shichao Xu, Yixuan Wang, Yanzhi Wang, Zheng O'Neill, and Qi Zhu. One for many: Transfer learning for building hvac control. In *Proceedings of the 7th ACM international conference on systems for energy-efficient buildings, cities, and transportation*, pages 230–239, 2020.

[60] Mengjiao Yang, Bo Dai, Ofir Nachum, George Tucker, and Dale Schuurmans. Offline policy selection under uncertainty. In *International Conference on Artificial Intelligence and Statistics*, pages 4376–4396. PMLR, 2022.

[61] Tao Yang, Arkasama Bandyopadhyay, Zheng O'Neill, Jin Wen, and Bing Dong. From occupants to occupants: A review of the occupant information understanding for building hvac occupant-centric control. In *Building simulation*, volume 15, pages 913–932. Springer, 2022.

[62] Chao Yu, Akash Velu, Eugene Vinitsky, Jiaxuan Gao, Yu Wang, Alexandre Bayen, and Yi Wu. The surprising effectiveness of ppo in cooperative multi-agent games. *Advances in Neural Information Processing Systems*, 35:24611–24624, 2022.

[63] 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.

[64] Tianyu Zhang and Omid Ardakanian. Cobs: Comprehensive building simulator. In *Proceedings of the 7th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation*, BuildSys '20, page 314–315, New York, NY, USA, 2020. Association for Computing Machinery.

[65] Tianyu Zhang, Gaby Baasch, Omid Ardakanian, and Ralph Evins. On the joint control of multiple building systems with reinforcement learning. In *Proceedings of the Twelfth ACM International Conference on Future Energy Systems*, e-Energy '21, page 60–72, New York, NY, USA, 2021. Association for Computing Machinery.

[66] Tianyu Zhang, Aakash Krishna GS, Mohammad Afshari, Petr Musilek, Matthew E Taylor, and Omid Ardakanian. Diversity for transfer in learning-based control of buildings. In *Proceedings of the Thirteenth ACM International Conference on Future Energy Systems*, pages 556–564, 2022.

[67] Peng Zhao, Siddharth Suryanarayanan, and Marcelo Godoy Simoes. An energy management system for building structures using a multi-agent decision-making control methodology. *IEEE transactions on industry applications*, 49(1):322–330, 2012.

[68] Abilmansur Zhumabekov, Daniel May, Tianyu Zhang, Aakash Krishna GS, Omid Ardakanian, and Matthew E Taylor. Ensembling diverse policies improves generalizability of reinforcement learning algorithms in continuous control tasks. In *Proceedings of the Adaptive and Learning Agents Workshop (ALA 2023)*. ALA, 2023.