Transactions on Control Systems Technology

Energy-Efficient Control of a Building HVAC System using Reinforcement Learning

Journal:	*IEEE Transactions on Control Systems Technology*
Manuscript ID	Draft
mstype:	Regular Paper
Date Submitted by the Author:	n/a
Complete List of Authors:	Raman, Naren Srivaths; University of Florida, Mechanical and Aerospace Engineering Devraj, Adithya M.; University of Florida, Electrical and Computer Engineering Barooah, Prabir; University of Florida, Department of Mechanical and Aerospace Engineering Meyn, Sean; University of Florida, Electrical and Computer Engineering
Keywords:	building HVAC system, reinforcement learning (RL), building climate controller, model predictive control (MPC)

Energy-Efficient Control of a Building HVAC System using Reinforcement Learning

Naren Srivaths Raman, Adithya M. Devraj, Prabir Barooah, and Sean P. Meyn

***Abstract*— Advanced climate control algorithms provide a cost-effective approach to reduce the large energy footprint of heating, ventilation, and air conditioning (HVAC) systems in buildings. Although model predictive control (MPC) has been extensively studied for this problem, it has not been widely adopted in practice. The main challenges of MPC are that it requires simple yet accurate models and complex real-time computation. Reinforcement learning (RL) is an attractive alternative to MPC: the real-time control computation is both model free and simple. However, RL has a computationally expensive learning process, and it has many design choices that affect performance.**

In this paper, we propose an RL controller for energy-efficient indoor climate control of commercial buildings. Unlike prior work on RL for HVAC systems, the proposed design is applicable to continuous actions and states. This feature makes the controller applicable to common HVAC system configurations. Extensive simulations are conducted under various scenarios to compare the performance of the proposed RL controller with that of an MPC controller and a common baseline (rule-based) controller. The performance of the RL controller is seen to be close to that of the MPC controller. Given the ease of implementation of RL compared to MPC, these results indicate RL is a strong competitor of MPC for HVAC control.

I. Introduction

Globally, about 20% of the total energy consumption is by buildings, and it is expected to grow by 1.3% per year on average from 2018 to 2050 [1]. A major portion of this energy is used by heating, ventilation, and air conditioning (HVAC) systems. The use of advanced climate control algorithms is a cost-effective way to reduce the large energy footprint of HVAC systems. This is especially so in commercial buildings: most modern commercial buildings are equipped with the required sensors, actuators, and communication infrastructure [2, 3].

A good climate control algorithm should be able to satisfy three main goals: (i) ensure thermal comfort, (ii) maintain good indoor air quality, and (iii) use the minimum amount of energy. Currently used control algorithms are rule-based and utilize conservatively designed set points, ensuring thermal comfort and indoor air quality, but can cause high energy consumption [4].

In recent years, model predictive control (MPC) has emerged in the literature as a popular alternative to rule-based control [2]. This is because MPC has the ability to satisfy all the competing goals mentioned above. Despite many simulation—and even some experimental—works showcasing the ability of MPC to substantially reduce energy usage, MPC has not been widely adopted in practice. A challenge with MPC is that simple yet accurate models are needed for real-time optimization. Obtaining such control-oriented models is a challenging task, particularly for HVAC systems, since the process dynamics are complex. Another challenge with MPC for buildings is the associated computational complexity. The optimization in MPC is typically non-convex because of the nonlinearities in the hygrothermal dynamics of the building and its HVAC system [5–7]. Depending on the planning horizon, the number of decision variables can be large, and solving such a high dimensional non-convex constrained optimization problem in real-time, at every time step, can be challenging.

An alternative approach to MPC is reinforcement learning (RL): a collection of tools used to approximate an optimal policy based on data gathered from a physical system or its simulation. It has two key advantages over MPC:

- It is "model free": no plant model is needed to compute the control decisions in real-time. Although a simulation model is needed for off-line learning, unlike MPC, this model can be complex since it is not used for on-line optimization.
- The real-time control computation required is quite simple since the RL controller is a state-feedback controller.

These benefits are accompanied with limitations as well. First, the speed and simplicity of on-line computation comes with computationally complex off-line learning. Second, state constraints cannot be directly specified to the controller, they can only be indirectly encouraged by appropriately designing the so-called reward/cost function. These design choices affect performance. HVAC control has strict state constraints: the primary function of the climate controller is to maintain indoor temperature and humidity within predetermined limits, energy efficiency is secondary. Third, popular RL techniques like Watkins' Q-learning [8] and deep Q-network (DQN) [9] lack theoretical convergence guarantees when control actions and states take values from uncountable sets. In case of building climate control, control actions (such as air flow rate) and states (such as indoor temperature) are continuous valued variables, and thus fall into this category.

In this paper, we propose and evaluate an RL controller for control of a commercial building HVAC system. Although application of RL to building climate control is gaining popularity in the literature, work on this topic is limited;

This research reported here has been partially supported by the NSF through awards # 1646229 (CPS/ECCS) and # 1934322 (CMMI).

The authors are with the University of Florida, Gainesville, Florida 32611, USA. narensraman@ufl.edu, adithyamdevraj@ufl.edu, pbarooah@ufl.edu, meyn@ece.ufl.edu.

see the review paper [10] and references therein. In [11], RL is used to optimally control active and passive thermal storage inventory in a commercial building, while in [12] RL is used to vary the cooling temperature setpoints of an office building. Both use Watkins' Q-learning algorithm. A version of "deep reinforcement learning" (DRL) is used to control radiant heating system in an office building in [13]. The controlled actions are on/off of a steam heat exchanger, and discretized supply water temperature set point. In [14], DRL is used to vary air flow rate supplied to zones in a building (among discrete levels), while [15] uses DRL for jointly controlling HVAC system and datacenter workloads in mixed-use buildings (buildings in which datacenters are physically co-located with office rooms). Ref. [16] uses DRL to optimally control the temperature set point of a zone in five discrete levels so that the thermal comfort of a group of occupants in the zone is maximized.

Contributions:

(i) All the prior works mentioned above either discretize both the state and action space or discretize just the action space. For the HVAC system configuration considered in this work, which has four control commands, even a coarse discretization of actions leads to a large number of possible combinations. We address the challenges of continuous state and action spaces by applying the Zap Q-learning algorithm [17] with linear function approximation; Zap-Q is the only Q-learning algorithm that is designed to converge in a general function approximation setting, *and* achieve optimal convergence rate [18].

(ii) We provide guidelines to address some of the design challenges in RL:

- The cost function is designed to reduce energy use while keeping the temperature and humidity within predetermined limits during implementation (even without access to a prediction model).
- The state-space is designed so that the actuator constraints are implicitly imposed when implementing the state-feedback controller.

(iii) The performance of the RL controller is compared with an MPC controller that is designed to minimize energy use while maintaining indoor climate constraints (in our prior work [7]) and a widely used rule-based controller called *single maximum* [19].

(iv) The robustness of the RL controller is evaluated by experimenting with different levels of occupancy in the plant and different outdoor weather conditions.

(v) Interpretations of the control policy induced by the Q-function learned by the RL algorithm are discussed.

Simulation results show that the proposed RL controller is able to maintain temperature and humidity within the comfort limits under most conditions, and reduces energy usage when compared to the baseline. The energy savings are comparable (only slightly lower) to those achieved by the MPC controller. This is remarkable, given the obvious advantages of RL: it is largely model free, and the on-line computational complexity is reduced by orders of magnitude. Further discussion can be found in the conclusion.

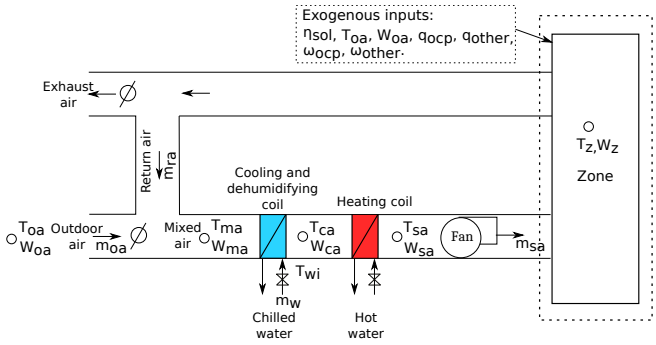


Fig. 1: Schematic of a single-zone commercial variable-air-volume HVAC system.

A preliminary version of this work was reported in [20], wherein the performance of the RL, MPC, and baseline controllers is compared. There are several differences between [20] and this paper. We have included a section in this paper that provides interpretations of the control policy learned by the RL controller (Section V-C). The baseline controller used in this paper is more energy-efficient than the one used in [20], which provides more confidence on the energy efficiency achieved by the RL controller. Finally, unlike [20], this paper contains a thorough investigation of the robustness to differences between scenarios used in learning and scenarios encountered in real-time control, which is essential to assess field-readiness.

The rest of this paper is organized as follows. Section II describes the HVAC system considered in this work and the models used in simulating the plant. Section III presents our proposed RL-based controller. Section IV describes the MPC and baseline controllers used for comparison. Simulation results are discussed in Section V. Section VI summarizes the conclusions.

II. System Description and Models

We consider a single-zone commercial building equipped with a variable-air-volume HVAC system, whose schematic is shown in Figure 1. In such a system, the outdoor air is mixed with part of the air exhausted from the zone. This mixed air is sent through the cooling coil where the air is cooled and dehumidified to conditioned air temperature (T_{ca}) and humidity ratio (W_{ca}). Then the conditioned air is reheated as needed and finally supplied to the zone. Note that there is only change in temperature across the reheat coil, the humidity remains constant, i.e., $W_{sa} = W_{ca}$, where W_{sa} is the humidity ratio of the supply air.

There are 4 control commands: (i) supply air flow rate (m_{sa}), (ii) outdoor air ratio ($r_{oa} := \frac{m_{oa}}{m_{sa}}$, where m_{oa} is the outdoor air flow rate), (iii) conditioned air temperature (T_{ca}), and (iv) supply air temperature (T_{sa}). So, the control command/input vector is:

$$u(t) := [m_{sa}(t), r_{oa}(t), T_{ca}(t), T_{sa}(t)]^T \in \Re^4. \quad (1)$$

These inputs need to be varied in such a way that the thermal comfort and indoor air quality are maintained in the zone, while using minimal energy.

In order to compare the performance of the control algorithms presented and to train the proposed RL controller, we need simulation models of the zone's temperature and humidity dynamics, cooling and dehumidification coil, heating coil, and HVAC system's power consumption. These models are presented in detail in our prior work [7]. We present only the relevant salient features below.

A. Temperature and Humidity Dynamic Model

The overall plant model consists of hygrothermal dynamics of a single-zone building coupled with that of a cooling and dehumidification coil, and a heating coil. The zone and the coils are interconnected as shown in Figure 1. A resistor-capacitor (R-C) model is used to model the thermal dynamics of the zone; specifically, a 2R-2C model with the two states being the zone temperature and wall temperature. The humidity dynamic is modeled based on mass balance.

The overall model is of the form:

$$\dot{x}(t) = f\big(x(t), u(t), w(t)\big), \tag{2}$$

where the state vector $x(t)$ consists of zone temperature (T_z), wall temperature (T_w), humidity ratio of the indoor air (W_z), and humidity ratio of the conditioned air downstream of the cooling coil before entering the building ($W_{ca}(t)$), i.e., $x(t) := [T_z(t), T_w(t), W_z(t), W_{ca}(t)]^T \in \Re^4$. The control command $u(t)$ is defined in (1). The exogenous input $w(t) \in \Re^7$ consists of (i) outdoor air temperature (T_{oa}), (ii) outdoor air humidity ratio (W_{oa}), (iii) solar irradiance (η_{sol}), (iv,v) internal heat load (q_{ocp}) and internal water vapor generation rate (ω_{ocp}) due to people, and (vi,vii) internal heat load (q_{other}) and internal water vapor generation rate (ω_{other}) due to other sources like equipment, i.e., $w(t) := [T_{oa}(t), W_{oa}(t), \eta_{sol}(t), q_{ocp}(t), \omega_{ocp}(t), q_{other}(t), \omega_{other}(t)]^T$.

B. Power Consumption Models

There are three major components which consume power in the HVAC system: fan, cooling coil, and reheat coil. We assume that the power consumed by other components such as damper motors is negligible.

The fan power consumption P_{fan} is modeled as quadratic in the supply air flow rate [21]. The cooling coil power consumption is modeled as proportional to the heat extracted from the mixed air stream: $P_{cc}(t) = g_{cc}\big(m_{sa}(t), h_{ma}(t), h_{ca}(t)\big)$, where h_{ma} is the enthalpy of the mixed air and h_{ca} is the enthalpy of the conditioned air.

The reheat coil power consumption is modeled as proportional to the heat added to the conditioned air stream: $P_{reheat}(t) = g_{reheat}\big(m_{sa}(t), T_{sa}(t), T_{ca}(t)\big)$.

III. Reinforcement Learning-Based Controller

A. Markov Decision Process (MDP) Formulation

Reinforcement learning (RL) is a collection of tools used to approximate an optimal policy based on data gathered from a physical system or its simulation. In our application, RL is used to learn a control policy that minimizes energy consumption of a commercial building HVAC system, while ensuring that the zone temperature and humidity are within the desired comfort limits.

To this end, we model the problem as a discrete-time Markov decision process comprised of the tuple $(\mathcal{X}, \mathcal{U}, \mathcal{P}, c, \beta)$, where $\mathcal{X}$ denotes the state-space, $\mathcal{U}$ denotes the action-space (a set of all *feasible* actions), $\mathcal{P}$ denotes the transition kernel, $c : \mathcal{X} \times \mathcal{U}(\mathcal{X}) \times \mathcal{X} \to \Re$ denotes the cost function, and β denotes a discount factor.

Since the underlying system described in Section II is evolving in continuous time, we discretize time for our MDP formulation, with a sampling interval $\Delta t > 0$. We denote by $X_k \in \mathcal{X}$ the state, and $U_k \in \mathcal{U}$ the control input at time-instant $k\Delta t$ (or simply, time-step k).

The goal is to obtain a state-feedback policy $\phi^* : \mathcal{X} \to \mathcal{U}$ that minimizes the sum of expected discounted cost:

$$\phi^* := \underset{\phi:\mathcal{X}\to\mathcal{U}}{\operatorname{argmin}} \left\{ \sum_{k=0}^{\infty} \beta^k \mathsf{E}\big[c(X_k, U_k, X_{k+1})\big] \right\} \tag{3}$$

with $U_k = \phi(X_k)$ for $k \geq 0$.

There are several design choices that need to be made when formulating the underlying problem as an MDP. Here, we discuss some of the choices we made in our formulation, and the rationale behind them.

Choosing the state-action space:

Based on the system model defined in Section II, we define:

$$U_k := [m_{sa}(k), r_{oa}(k), T_{ca}(k), T_{sa}(k)]^T. \tag{4}$$

The choice of X_k is more subtle. The notion of "state" is flexible in both control theory [22] and reinforcement learning [23]. The original motivation is the same in each field: for the purposes of on-line decision making, replace the full history of observations at time k by some finite dimensional "sufficient statistic" X_k. One constraint that arises in RL is that the state process must be directly observable; in particular, the belief state that arises in partially observed MDPs requires the (model based) nonlinear filter, and is hence not directly useful for model-free RL. In practice, the "RL state" is specified as some compression of the full history of observations. The reader is referred to [23, Section 17.3] for further discussion. In this paper, as in many others, the "RL state" will be based on an intuitive selection of variables: present and recent observations.

Based on the discussion in Section II-A, it is natural to include the observables zone temperature $T_z(k)$ and zone humidity $W_z(k)$ to be a part of X_k. In addition, we also include certain exogenous inputs, as they can provide valuable information for decision-making[1]. Among the various exogenous inputs defined in Section II-A, only outdoor air temperature T_{oa}, outdoor air humidity W_{oa}, and solar

[1]See Section V-C for details on how this can help the controller make use of the latent heat load information to make "good" decisions.

irradiance η_{sol} are measured. Of these, we choose to include T_{oa} and W_{oa} in the state for the following reasons:

- Since solar irradiance is highly correlated with the outdoor air temperature, making it a part of the system state does not provide much additional useful information.
- In many cases, buildings are not equipped with a solar irradiance sensor.

Lastly, we include U_{k-1} in X_k: the control inputs in the previous time-step. This is required to impose rate constraints in a state-feedback policy. In conclusion:

$$X_k := [T_z(k), W_z(k), T_{oa}(k), W_{oa}(k), m_{sa}(k-1), r_{oa}(k-1), T_{ca}(k-1), T_{sa}(k-1)]^T \quad (5)$$

Feasible actions/control inputs:

At each time-step k, the input U_k needs to satisfy the following constraints that are imposed due to actuator limits, ventilation requirements, etc.

$$\max\left(m_{sa}(k-1) - m_{sa}^{rate}\Delta t, m_{sa}^{low}\right) \le m_{sa}(k) \le \min\left(m_{sa}(k-1) + m_{sa}^{rate}\Delta t, m_{sa}^{high}\right) \quad (6a)$$

$$\max\left(r_{oa}(k-1) - r_{oa}^{rate}\Delta t, r_{oa}^{low}\right) \le r_{oa}(k) \le \min\left(r_{oa}(k-1) + r_{oa}^{rate}\Delta t, r_{oa}^{high}\right) \quad (6b)$$

$$\max\left(T_{ca}(k-1) - T_{ca}^{rate}\Delta t, T_{ca}^{low}\right) \le T_{ca}(k) \le \min\left(T_{ca}(k-1) + T_{ca}^{rate}\Delta t, T_{ma}(k)\right) \quad (6c)$$

$$\max\left(T_{sa}(k-1) - T_{sa}^{rate}\Delta t, T_{ca}(k)\right) \le T_{sa}(k) \le \min\left(T_{sa}(k-1) + T_{sa}^{rate}\Delta t, T_{sa}^{high}\right) \quad (6d)$$

If U_k satisfies the above constraints, we say $U_k \in \mathcal{U}(X_k)$.

Constraint (6a) is imposed to take into account the capabilities of the fan. The minimum allowed value for the supply air flow rate is computed based on the ventilation requirements specified in ASHRAE 62.1 [24], as well as to maintain positive building pressurization. This is computed as follows: $m_{sa}^{low} = \max\left\{(m_{oa}^p n_p + m_{oa}^A A)/r_{oa}, m_{oa}^{bp}/r_{oa}\right\}$, where m_{oa}^p is the outdoor air flow rate required per person, n_p is the number of people, m_{oa}^A is the outdoor air required per zone area, A is the zone area, m_{oa}^{bp} is the outdoor air flow rate required to maintain positive building pressurization, and r_{oa} is the outdoor air ratio.

Constraints (6b)-(6d) take into account the capabilities of the damper actuators, cooling and reheat coils. In constraint (6c), T_{ma} is the mixed air temperature computed as: $T_{ma}(k) = r_{oa}(k)T_{oa}(k) + (1 - r_{oa}(k))T_z(k)$. The inequality $T_{ca}(k) \le T_{ma}(k)$ ensures that the cooling coil can only cool the mixed air stream. Similarly, the inequality $T_{sa}(k) \ge T_{ca}(k)$ ensures that the reheat coil can only add heat.

Cost function design:

A climate control system has to keep overall energy consumption low while maintaining temperature and humidity of the zone within a desired range. The cost function is designed to capture both these features.

Denote by $\mathcal{X}_{\text{de}}$ the set of all desirable states: at time step $k \ge 0$, we say $X_k \in \mathcal{X}_{\text{de}}$, if the zone temperature and humidity are within desirable limits: $T_z^{low} \le T_z(k) \le T_z^{high}$ and $W_z^{low} \le W_z(k) \le W_z^{high}$, for given values of T_z^{low} T_z^{high}, W_z^{low}, and W_z^{high}. The cost function is then defined as,

$$c(X_k, U_k, X_{k+1}) = \begin{cases} E_k, & \text{if } X_{k+1} \in \mathcal{X}_{\text{de}} \\ p_{low}, & \text{if } X_k \in \mathcal{X}_{\text{de}} \ \& \ X_{k+1} \notin \mathcal{X}_{\text{de}} \\ p_{high}, & \text{if } X_k \notin \mathcal{X}_{\text{de}} \ \& \ X_{k+1} \notin \mathcal{X}_{\text{de}} \end{cases}$$

where E_k denotes the energy consumed:

$$E_k := E_{fan}(k) + E_{cc}(k) + E_{reheat}(k), \quad (7)$$

where $E_{fan}(k)$, $E_{cc}(k)$, $E_{reheat}(k)$ are the energy consumed by the fan, cooling coil, and reheat coil, respectively, and $p_{high} > p_{low}$ are the penalties imposed for violating the comfort limits. In Section V-C we discuss how adding such penalties can lead to a model-free approach to implicitly impose state-constraints.

An occasional violation of limits on T_z or W_z may not be noticed by occupants, whereas a sustained violation can cause discomfort and even lead to serious adverse effects such as mold. To discourage this behavior, we impose the higher penalty p_{high} ($> p_{low}$) in the cost $c(\cdot)$ when temperature/humidity constraints are violated over two consecutive time steps.

B. Value Function Approximation and Zap Q-Learning

Under the assumption that the underlying problem is an MDP, it is known that the optimal policy in (3) satisfies:

$$\phi^*(x) = \operatorname*{argmin}_{u \in \mathcal{U}(x)} Q^*(x,u), \qquad x \in \mathcal{X} \quad (8)$$

where $Q^* : \mathcal{X} \times \mathcal{U}(\mathcal{X}) \to \Re$ denotes the associated optimal Q-function:

$$Q^*(x,u) := \min_{\{U_k\}} \sum_{k=0}^{\infty} \beta^k \mathsf{E}\left[c(X_k, U_k, X_{k+1}) | X_0 = x, U_0 = u\right]$$

where the minimization is over all feasible inputs. The function Q^* solves the Bellman equation:

$$\begin{aligned} Q^*(x,u) = E\Big[c(X_k, U_k, X_{k+1}) \mid X_k = x, U_k = u\Big] \\ + \beta E\Big[\underline{Q}^*(X_{k+1}) \mid X_k = x, U_k = u\Big] \end{aligned} \quad (9)$$

where, for any $Q : \mathcal{X} \times \mathcal{U}(\mathcal{X}) \to \Re$, we use the notation:

$$\underline{Q}(x) := \min_{u \in \mathcal{U}(x)} Q(x,u).$$

We apply the Zap Q-learning of [17] to approximate Q^* using a parameterized family of functions $\{Q^\theta : \theta \in \Re^d\}$. Specifically, we consider linear parameterization, so that for each $x \in \mathcal{X}$ and $u \in \mathcal{U}(\mathcal{X})$,

$$Q^\theta(x,u) = \theta^T \psi(x,u), \quad (10)$$

where $\psi : \mathcal{X} \times \mathcal{U}(\mathcal{X}) \to \Re^d$ denotes the "basis functions". We choose the basis functions to be a quadratic in (x,u): the components of the vector $\psi(\cdot)$ are shown in Table I. Once the basis functions are fixed, the Zap Q-learning

algorithm defined below can be used to estimate Q^* using the approximation Q^{θ^*}.

Algorithm 1 Zap Q-learning with ε-greedy exploration

Input: $\theta_0 \in \Re^d$, $\hat{A}_0 \in \Re^{d\times d}$, $X_0 \in \mathcal{X}$, $k=0$, $T \in \mathcal{Z}^+$, $\rho \in (0.5,1)$, $\varepsilon \in (0,1)$

1: **repeat**

2: With prob. ε, choose $U_k \in \mathcal{U}(X_k)$ uniformly at random, and with prob. $1-\varepsilon$, choose: $U_k = \underset{u\in\mathcal{U}(X_k)}{\operatorname{argmin}} Q^{\theta_k}(X_k,u)$

3: Sample next state X_{k+1} with current state X_k and input U_k

4: $\phi_k(X_{k+1}) := \underset{u\in\mathcal{U}(X_{k+1})}{\operatorname{argmin}} Q^{\theta_k}(X_{k+1},u)$;

5: $d_{k+1} := c(X_k,U_k,X_{k+1}) + \beta Q^{\theta_k}(X_{k+1},\phi_k(X_{k+1})) - Q^{\theta_k}(X_k,U_k)$; ▷ Temporal difference

6: $A_{k+1} := \psi(X_k,U_k)\big[\beta\psi(X_{k+1},\phi_k(X_{k+1})) - \psi(X_k,U_k)\big]^T$;

7: $\hat{A}_{k+1} = \hat{A}_k + \gamma_k\big[A_{k+1} - \hat{A}_k\big]$; ▷ Matrix gain update

8: $\theta_{k+1} = \theta_k - \alpha_k \hat{A}_{k+1}^{-1}\psi(X_k,U_k)d_{k+1}$; ▷ Zap-Q update

9: $k = k+1$

10: **until** $k \ge T$

The step-size sequences $\{\alpha_k\}$ and $\{\gamma_k\}$ are chosen to be:

$$\alpha_k = (k+1)^{-1} \quad \text{and} \quad \gamma_k = \alpha_k^{\rho}, \qquad k \ge 0$$

where $\rho \in (0.5,1)$ is a hyper-parameter. The input sequence defined in line 2 is known as "ϵ-greedy" exploration. The matrix $\hat{A}_k$ in Algorithm 1 may not be invertible in initial iterations. We therefore warm start the training by applying the *Kalman filter* matrix gain algorithm of [25] (which is well-defined for each $k \ge 0$) for the first T_{ws} iterations. The warm-start parameter $T_{\text{ws}} \ll T$ is a design choice.

The use of Zap-Q with linear function approximation (specifically, the quadratic basis) has several advantages over existing techniques:

(i) Contrary to existing Q-learning algorithms, it is shown in [18] that Zap-Q converges under very general assumptions, *even in a non-linear function approximation setting*. The proof is based on approximating the evolution of the parameters by a globally convergent *Newton-Raphson flow* [26, 27].

(ii) Algorithms such as DQN [9], though they use neural-networks to approximate the Q-function, implementation in continuous action-space setting can be hard; this is due to the fact that minimization of a continuous function of the actions (a necessary step in the Q-learning algorithm), when the function is the output of a neural network, can be computationally very expensive [13–16]. Our formulation on the other hand does not have these issues.

(iii) Under mild conditions, Zap-Q algorithm is known to be the fastest converging Q-learning algorithm [17, 18, 28].

C. Dealing with Occupied and Unoccupied Modes

The constraints on zone temperature (T_z) are typically more relaxed during the unoccupied mode when compared to the occupied mode: $[T_z^{low,occ}, T_z^{high,occ}] \subseteq [T_z^{low,unocc}, T_z^{high,unocc}]$. Suppose that at a given time k, $T_z(k)$ is between the higher limit of the occupied mode ($T_z^{high,occ}$) and the higher limit of the unoccupied mode ($T_z^{high,unocc}$). If the building is in the occupied mode, then the zone is clearly violating the temperature constraints, but if the building is in the unoccupied mode, the zone is within the constraints. This suggests that we learn different control policies for each of the two, occupied and unoccupied modes. To enable such distinct control decisions, we use different set of Q-function parameters for the occupied and unoccupied modes: The total number of parameters (θ) in the Q function is $2d = 2\times 45 = 90$.

TABLE I: Basis functions $\psi_i^k := \psi_i(X_k,U_k)$ at iteration k.

$\psi_1^k = 1$	$\psi_2^k = T_z(k)$	$\psi_3^k = W_z(k)$
$\psi_4^k = T_{oa}(k)$	$\psi_5^k = W_{oa}(k)$	$\psi_6^k = m_{sa}(k)$
$\psi_7^k = r_{oa}(k)$	$\psi_8^k = T_{ca}(k)$	$\psi_9^k = T_{sa}(k)$
$\psi_{10}^k = (\psi_2^k)^2$	$\psi_{11}^k = (\psi_3^k)^2$	$\psi_{12}^k = (\psi_4^k)^2$
$\psi_{13}^k = (\psi_5^k)^2$	$\psi_{14}^k = (\psi_6^k)^2$	$\psi_{15}^k = (\psi_7^k)^2$
$\psi_{16}^k = (\psi_8^k)^2$	$\psi_{17}^k = (\psi_9^k)^2$	$\psi_{18}^k = \psi_2^k\psi_3^k$
$\psi_{19}^k = \psi_2^k\psi_4^k$	$\psi_{20}^k = \psi_2^k\psi_5^k$	$\psi_{21}^k = \psi_2^k\psi_6^k$
$\psi_{22}^k = \psi_2^k\psi_7^k$	$\psi_{23}^k = \psi_2^k\psi_8^k$	$\psi_{24}^k = \psi_2^k\psi_9^k$
$\psi_{25}^k = \psi_3^k\psi_4^k$	$\psi_{26}^k = \psi_3^k\psi_5^k$	$\psi_{27}^k = \psi_3^k\psi_6^k$
$\psi_{28}^k = \psi_3^k\psi_7^k$	$\psi_{29}^k = \psi_3^k\psi_8^k$	$\psi_{30}^k = \psi_3^k\psi_9^k$
$\psi_{31}^k = \psi_4^k\psi_5^k$	$\psi_{32}^k = \psi_4^k\psi_6^k$	$\psi_{33}^k = \psi_4^k\psi_7^k$
$\psi_{34}^k = \psi_4^k\psi_8^k$	$\psi_{35}^k = \psi_4^k\psi_9^k$	$\psi_{36}^k = \psi_5^k\psi_6^k$
$\psi_{37}^k = \psi_5^k\psi_7^k$	$\psi_{38}^k = \psi_5^k\psi_8^k$	$\psi_{39}^k = \psi_5^k\psi_9^k$
$\psi_{40}^k = \psi_6^k\psi_7^k$	$\psi_{41}^k = \psi_6^k\psi_8^k$	$\psi_{42}^k = \psi_6^k\psi_9^k$
$\psi_{43}^k = \psi_7^k\psi_8^k$	$\psi_{44}^k = \psi_7^k\psi_9^k$	$\psi_{45}^k = \psi_8^k\psi_9^k$

D. Off-Line Learning and Real-Time Control

Algorithm 1 is implemented using a simulation model of the building to learn the optimal policy. Once the optimal Q-function—rather its approximation—is learned, the online state-feedback control is obtained using (8), with Q^* replaced by Q^{θ_T}, where θ_T denotes the parameters learned with Algorithm 1, after T iterations. That is, real-time control command is computed as:

$$U_k = \phi^{\theta_T}(X_k) := \underset{u\in\mathcal{U}(X_k)}{\operatorname{argmin}} Q^{\theta_T}(X_k,u)\,. \tag{11}$$

Notice that the optimization problem involved in real-time control computation (11) is straightforward to solve. From (10) and Table I we can see that the cost function is quadratic in u, with box constraints (6a)-(6d), and just 4 decision variables.

IV. Control Algorithms Used for Comparison

We compare the performance of the RL controller with two others, an MPC controller and a rule-based controller. These are briefly described below. The interested reader is referred to [7] for a detailed description of the MPC controller.

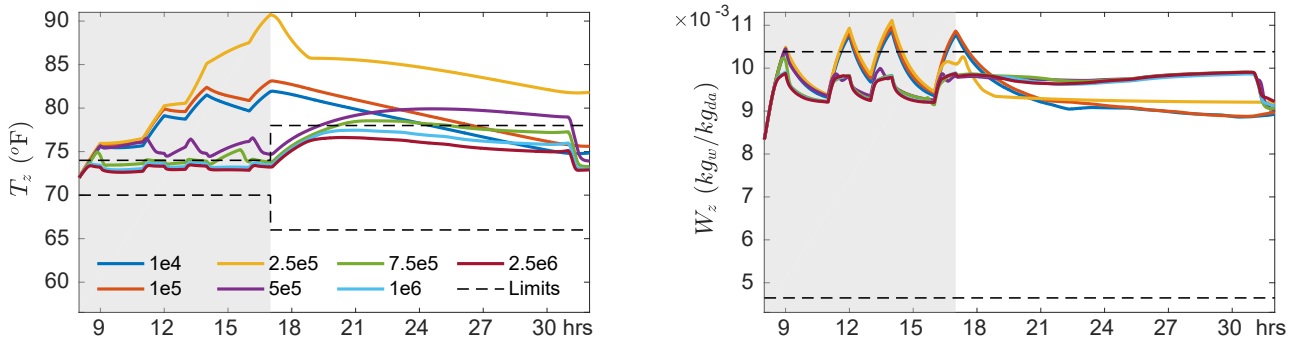


Fig. 2: Closed-loop response obtained using the parameters (θ) learned after different number of iterations for the RL controller. Left: zone air temperature. Right: zone air humidity ratio. There are no temperature or humidity violations after $1e6$ iterations. The scheduled hours of occupancy are shown as the gray shaded area.

A. Model Predictive Controller

This controller is a slightly modified version of the MPC controller proposed in our prior work [7]. The goal of this MPC controller is the same as the proposed RL controller: minimize energy use while maintaining temperature and humidity constraints. The control inputs are computed in discrete time steps Δt, for a finite planning horizon N, by solving a constrained optimization problem. There are three pieces of information needed for solving this problem. They are: (i) the current value of the states, (ii) prediction of the exogenous inputs mentioned in Section II-A over the planning horizon, and (iii) model of the building and HVAC system. The control inputs for the first time step, obtained from solving this problem, are applied to the plant. At the next time step this process is repeated.

The states for this MPC controller are the zone temperature and humidity, $x(k) = [T_z(k), W_z(k)]^T$. The control inputs are the same as those mentioned in Section II. The optimization problem solved is to minimize the total energy consumption—fan, cooling coil, and reheat coil—over a planning horizon N, subject to: (i) the various control input constraints—(6a)-(6d)—mentioned in Section III, (ii) equality constraints due to model of the temperature and humidity dynamics of the zone, (iii) box constraints to maintain temperature and humidity of the zone within the allowed comfort limits, with slack variables for feasibility, (iv) equality constraints due to model of the cooling and dehumidifying coil. *Unlike RL, state constraints are explicitly imposed, as MPC has models to compute the state evolution/trajectory.*

B. Baseline Controller

For the baseline, we consider the widely used rule-based controller called *single maximum* [19]. This controller operates in three modes based on the zone temperature: cooling, deadband, and reheating. The supply air flow rate (m_{sa}) and supply air temperature (T_{sa}) are varied based on the mode the controller is in. The minimum allowed value for the supply air flow rate is typically high so that the controller is capable of maintaining the design heat load with a low enough supply air temperature to prevent stratification.

The conditioned air temperature (T_{ca}) is typically kept constant at a low value ($55°F$), which ensures that the air supplied to the zone is dry enough at all times. The outdoor air ratio (r_{oa}) is varied to maintain the ventilation requirements dictated by ASHRAE 62.1 [24] and the positive building pressurization requirements.

The zone temperature set points vary based on occupied/unoccupied hours. The minimum allowed value for the supply air flow rate is reduced to meet just the building pressurization requirements when the controller is in the deadband mode during unoccupied hours; this prevents any unnecessary energy consumption.

V. Simulation Results and Discussions

A. Simulation Parameters

The parameters for the building and HVAC system models, presented in Section II, are chosen to mimic an auditorium in Pugh hall (5000 sq.ft.), located in the University of Florida campus. We present only the relevant details here, the interested readers are referred to [29] for a complete list of the parameter values used.

The scheduled hours of occupancy are 8:00 AM to 5:00 PM, during which the following temperature and humidity constraints are used: $T_z^{low,occ} = 294.3\ K\ (70\ °F)$, $T_z^{high,occ} = 296.5\ K\ (74\ °F)$, $W_z^{low,occ} = 0.0046\ kg_w/kg_{da}$, and $W_z^{high,occ} = 0.0104\ kg_w/kg_{da}$. The unoccupied hours are between 5:00 PM to 8:00 AM, during which the constraints are relaxed to: $T_z^{low,unocc} = 292\ K\ (66\ °F)$, $T_z^{high,unocc} = 298.7\ K\ (78\ °F)$, $W_z^{low,unocc} = 0.0046\ kg_w/kg_{da}$, and $W_z^{high,unocc} = 0.0104\ kg_w/kg_{da}$.

RL parameters: The various parameters used in the RL controller are listed in Tables II and III. MATLAB is used to run simulations to learn the parameters of the Q function. Three months of summer weather data, between June/06 to September/05 of 2016, for Gainesville, Florida, is obtained from the National Solar Radiation Database (`nsrdb.nrel.gov`). The parameters of the Q function for the occupied and unoccupied modes are obtained through

TABLE II: RL parameters.

β	ρ	p_{low}	p_{high}	T_{ws}
0.95	0.8	15	20	2×10^5

TABLE III: Parameters used in the RL and MPC controllers.

m_{sa}^{rate}	T_{ca}^{rate}	T_{sa}^{rate}	r_{oa}^{rate}
$(kg/s/min)$	(K/min)	(K/min)	$(\%/min)$
0.37	0.56	0.56	6
m_{sa}^{high}	T_{ca}^{low}	T_{sa}^{high}	r_{oa}^{low}
(kg/s)	(K)	(K)	$(\%)$
4.6	285.9	303.2	0
m_{oa}^{p}	m_{oa}^{A}	m_{oa}^{bp}	r_{oa}^{high}
$(kg/s/person)$	$(kg/s/m^2)$	(kg/s)	$(\%)$
0.0043	3.67×10^{-4}	0.1894	100

separate simulations as follows: We use 100% design occupancy of 175 people (n_p^{design}) for the implementation of Algorithm 1 in the occupied mode. For the unoccupied mode, zone has no occupants in our application of Algorithm 1. For performance evaluation, a different weather data from 2017 is used; see Section V-B for details.

The number of iterations T used before learning is stopped is chosen to be $T = 2.5 \times 10^6$. This value is obtained by examining the closed loop performance of the policy learned after a specific number of iterations T, and repeating it with increasing T, until acceptable performance—meaning consistent satisfaction of temperature and humidity constraints—is observed. Figure 2 shows the closed-loop responses obtained during this process; each curve corresponds to a distinct RL controller, i.e., distinct policy, differing in the amount of learning used to obtain the policy.

MPC parameters: We used a planning horizon of 24 hours, with sampling period of 5 minutes, resulting in $N = 288$. The underlying optimization problem has 3456 (=288×12) decision variables. The rate constraint values are the same as those presented in Table III for the RL controller.

Baseline controller parameters: The cooling and reheating set points are chosen to be 296.5 K (74 $^\circ F$) and 294.3 K (70 $^\circ F$) respectively, during the occupied mode. For the unoccupied mode, the cooling and reheating set points are 298.7 K (78 $^\circ F$) and 292 K (66 $^\circ F$) respectively. The conditioned air temperature is kept at 285.9 K (55 $^\circ F$).

Computational complexity: The optimization problem in RL and MPC is solved using CasADi [30] and IPOPT [31], a nonlinear programming (NLP) solver, on a Desktop computer running Linux with 16 GB RAM and a 3.60GHz × 8 CPU. Note that the number of decision variables for RL is only 4 while for MPC it is 3456. On an average it takes 0.1 seconds to solve the optimization problem for RL and 3 seconds for MPC. The parameters of the Q function are seen to settle after 2.5×10^6 iterations and takes about 42 hours to train.

B. Results and Discussions

We now compare the performance of the controllers through simulations. Outdoor weather data for Gainesville, Florida, obtained from National Solar Radiation Database (nsrdb.nrel.gov) is used. Simulations are done for different levels of occupancy (number of people) in the plant and different outdoor weather conditions to test the robustness of the RL controller; these are discussed in detail below.

1) Robustness to different levels of occupancy: Typically, in auditoriums/lecture classrooms, the actual occupancy level is lower than the design value. We test the performance of the RL controller, the MPC controller, and the baseline controller under such conditions. This is to examine if the controllers are robust to mismatch between the design conditions and reality. Four levels of occupancy are tested: 25%, 50%, 75%, and 100%. The simulations are done for 24 hours starting from 8:00 AM. The occupancy pattern considered in the plant is shown in Figure 3(a).

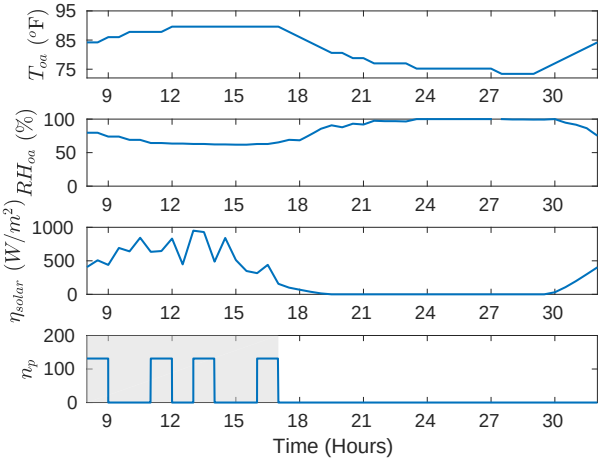
For each of the three controllers we assume the following: (i) the number of occupants is not measured, (ii) the zone is occupied throughout the scheduled hours of occupancy, between 8:00 AM to 5:00 PM, shown as the gray shaded area in Figure 3. So the controllers need to ensure that the outdoor air needed to satisfy the ventilation requirements corresponding to the designed number of occupants (175), is provided during these scheduled hours of occupancy, according to ASHRAE 62.1.

The MPC controller requires prediction of the exogenous inputs for the planning horizon N. We compute the occupant induced heat load (q_{ocp}) and rate of internal water vapor generation due to people (ω_{ocp}), based on the above mentioned occupancy pattern and designed number of occupants (175). The remaining exogenous inputs are assumed fully known.

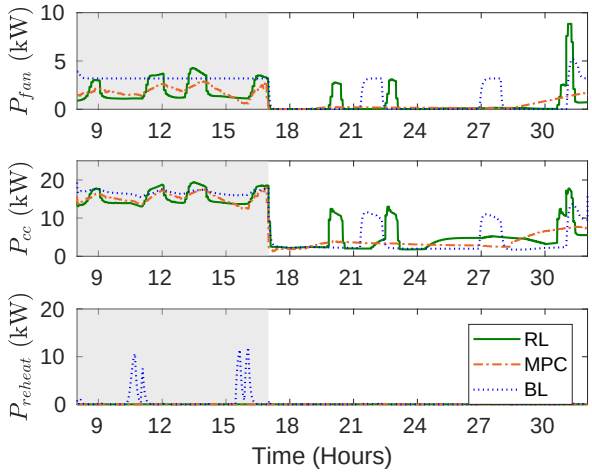
Figure 4 shows a comparison of the total energy consumed for a hot-humid day (July/03/2017) when using the three controllers for different levels of occupancy. The proposed RL controller saves 5 to 20% energy when compared to the baseline, while the MPC controller saves slightly more, 13 to 26%.

Figure 3 shows the simulation results for a hot-humid day, Jul/03/2017, with 75% occupancy level. The baseline controller sometimes leads to simultaneous heating and cooling, which can be seen between 10:00-11:00 hours and 15:00-16:00 hours in Figures 3(b) and 3(d). Such a phenomenon does not occur with the RL controller, which leads to energy savings; see Figure 3(b). As the occupancy level in the plant deviates from the design conditions (100%), this effect is more prominent, leading to higher energy savings by the RL controller as shown in Figure 4.

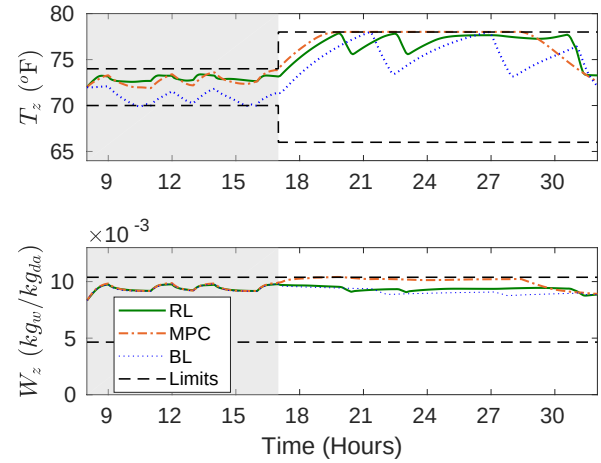
It can be seen from Figure 3(c) that both the RL and MPC controllers are able to maintain zone temperature and humidity within the comfort limits (shown as the black dashed lines). Note that for the MPC controller, the temperature and humidity constraints are active most of the time; see 19:00-29:00 hours in Figure 3(c). This behavior can be interpreted as the MPC controller being aggressive: it is trying to keep the zone as warm and humid as allowed so as to save energy, since it is a hot and humid day. Such behavior is not observed in the RL controller; it is less aggressive than the MPC controller. Since the RL controller is model-free, the state constraints cannot be explicitly imposed as in the case of



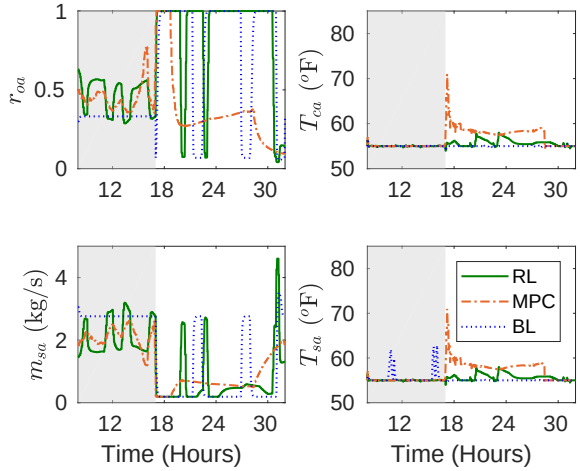
(a) Outdoor weather data and occupancy profile used (outdoor air temperature, outdoor air relative humidity, solar irradiance, and number of people).



(b) Comparison of the power consumptions (fan, cooling, and reheat power).



(c) Zone conditions with the black dashed lines showing the upper and lower comfort limits (zone air temperature and zone air humidity ratio).



(d) Control commands/inputs (outdoor air ratio, conditioned air temperature, supply air flow rate, and supply air temperature).

Fig. 3: Comparison of RL, MPC and baseline (BL) controllers for a hot-humid day (July/03/2017 starting from 8:00 AM) and 75% occupancy level. The scheduled hours of occupancy are shown as the gray shaded area.

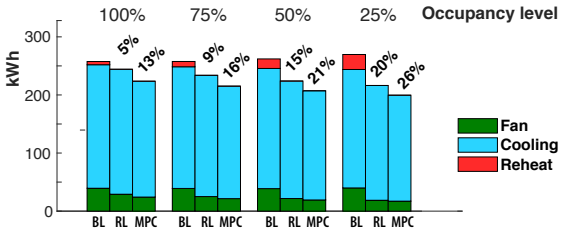


Fig. 4: Comparison of the total energy consumed over 24 hours by baseline (BL), RL, and MPC controllers for a hot-humid day (July/03/2017) and different levels of occupancy.

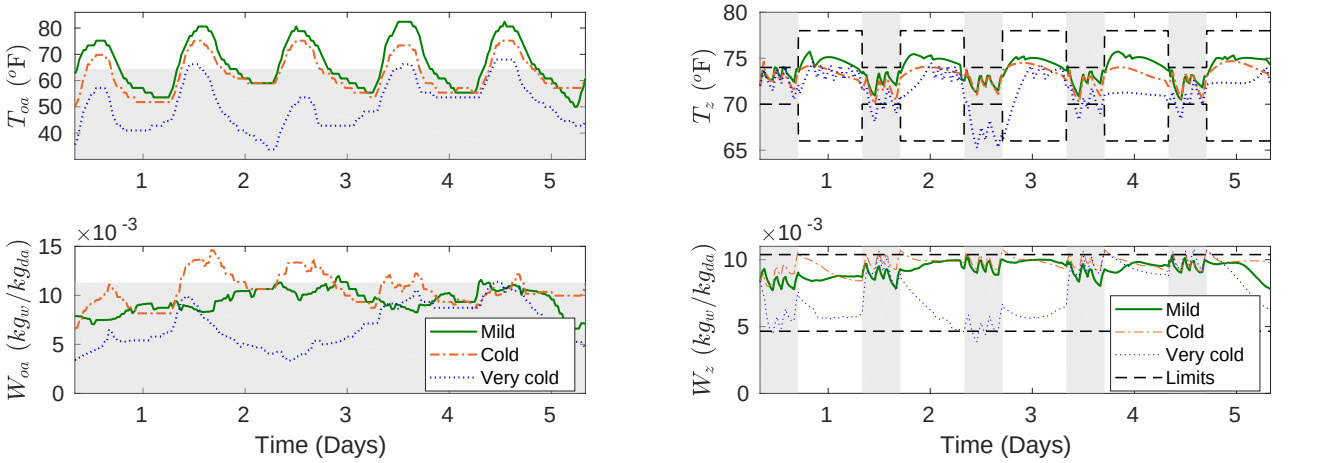
MPC. Rather it is learned from the penalties imposed due to state constraint violation during the learning process. This could be one of the reasons why the RL controller does not save as much energy as MPC.

The RL controller seems to have learned the state constraints, i.e., thermal comfort constraints, well. It can be seen from Figures 3(c) and 3(d) that as the zone temperature gets very close to the boundary—see around 20:00 hours and 23:00 hours—the supply airflow rate is increased and thus the zone temperature gets pushed back into its allowed limits.

The RL controller is found to be robust to lack of occupancy information. It is able to maintain the temperature and humidity at all four levels of occupancy even though it is trained assuming 100% occupancy.

2) Robustness to different outdoor weather conditions: We now test the performance of the RL controller under different outdoor weather conditions. Three kinds of weather conditions are considered: mild, cold, and very cold. Figure 5 shows the simulation results when using the RL controller for the three weather conditions. The simulations are done for 5 days starting from 8:00 AM.

The RL controller is able to maintain the zone temperature and humidity under mild and cold weather conditions almost always; see Figure 5(b). It is only under very cold weather condition that the RL controller is unable to maintain the temperature and humidity constraints; see between days 2 and 3 in Figure 5(b). We conjecture that the reason for



(a) Outdoor weather data (outdoor air temperature and outdoor air humidity ratio). The gray shaded area is the region that is never explored during the learning phase.

(b) Zone conditions with the black dashed lines showing the upper and lower comfort limits (zone air temperature and zone air humidity ratio). The scheduled hours of occupancy are shown as the gray shaded area.

Fig. 5: Performance comparison of the RL controller under different outdoor weather conditions. Mild: Mar/06/2017 to Mar/10/2017. Cold: Nov/27/2017 to Dec/01/2017. Very cold: Dec/11/2017 to Dec/15/2017. Outdoor weather data obtained from nsrdb.nrel.gov for Gainesville, FL.

this is the drastically different weather condition used in the testing phase when compared to the learning phase. Recall that summer weather data is used for learning the parameters of the RL controller as discussed in Section V-A. The conditions under which the RL controller is unable to maintain constraints—outdoor air temperature less than 53.6 $^\circ F$ and the outdoor air humidity ratio less than 0.0054 kg_w/kg_{da}—are conditions that the controller has never explored during the learning phase. The region that is never explored is shown as the gray shaded area in Figure 5(a).

C. Under The Hood

In this section we take a closer look at the policy induced by the Q function learned by the RL algorithm. Our goal is to obtain insights into the policy learned by the controller.

Including zone humidity in the state: Contrary to much of the previous literature [11, 14, 15], we include zone humidity in the state, as optimal control decisions taken in the absence of humidity can lead to higher energy use or humidity constraint violation, especially in hot and humid climates [29]. Figure 6 shows the optimal conditioned air temperature T_{ca} chosen by the RL controller under two different conditions: zone air is dry and zone air is humid. Except for the zone humidity ratio W_z being different, all other conditions are the same. The RL controller chooses a lower value for the conditioned air temperature T_{ca}—shown as the red dots—when the zone air is humid. This ensures that dryer air is supplied to the zone. If humidity is excluded from the state, such behavior would not be possible.

Including outdoor air humidity in the state: Including W_{oa} (an exogenous input) to be part of the state helps the controller to be aware of the latent cooling load while making decisions. For example, Figure 7 shows the optimal outdoor air ratio r_{oa}—shown as the red dots—chosen by the RL controller under two different conditions: outdoor weather is dry and outdoor weather is humid. It can be seen that the RL controller reduces the outdoor air ratio when the weather is humid, so that latent load of the cooling coil is reduced, thus reducing energy consumption.

Penalties for temperature and humidity constraints: We use large penalties in the cost function—see Section III-A—when the constraint on zone temperature (T_z) or zone humidity ratio (W_z) is violated during the learning phase. It is found that this did ensure that the RL controller kept T_z and W_z within the allowed limits under most conditions.

Figure 8 shows the optimal supply air flow rates—shown as the red dots—chosen by the RL controller under two conditions: zone air temperature at 72$^\circ F$ and at 74$^\circ F$. It can be seen that as T_z gets closer to the high limit (74$^\circ F$ during the occupied mode), the air flow rate is increased to cool down the zone. Note that a lower energy solution would be to let the zone get warm and not increase the air flow rate, since the outdoor condition used is hot ($T_{oa} = 80^\circ F$) and humid ($RH_{oa} = 50\%$). The controller does not do so

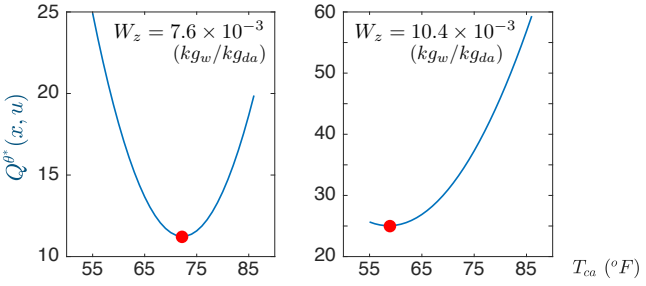


Fig. 6: RL controller chooses a lower conditioned air temperature for a more humid zone. Left: $W_z = 0.0076\ kg_w/kg_{da}$ (low). Right: $W_z = 0.0104\ kg_w/kg_{da}$ (high). Optimal control input chosen by the RL controller shown as the red dots.

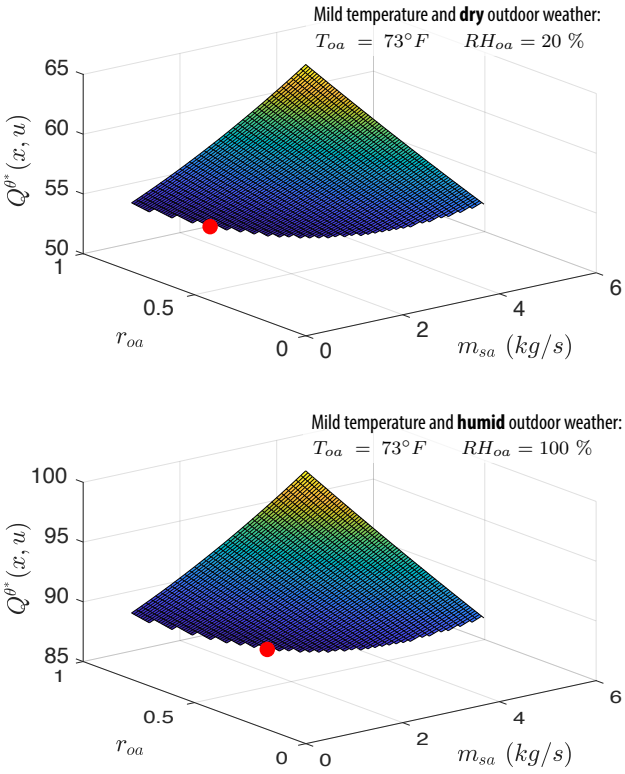


Fig. 7: RL controller chooses a lower outdoor air ratio when the weather is more humid to reduce the latent load of the cooling coil. Optimal control input chosen by the RL controller shown as the red dots.

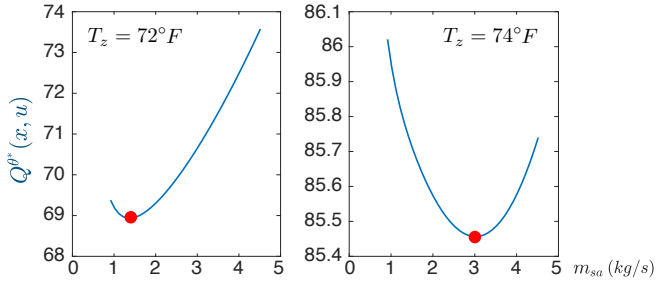


Fig. 8: RL controller chooses a higher supply air flow rate when the zone is warmer. Left: $T_z = 72\ ^\circ F$. Right: $T_z = 74\ ^\circ F$. Optimal control input chosen by the RL controller shown as the red dots.

because of the penalty imposed during the learning phase.

VI. Conclusion

The proposed RL controller performs nearly as well as the MPC controller. The main performance criterion for HVAC control is maintenance of indoor climate. As long as the weather conditions are not drastically different from the conditions explored during the learning phase, the RL controller maintains temperature and humidity within allowable limits just as the MPC and the baseline controllers do. The second criterion is energy savings, and though both RL and MPC provide substantial energy savings over baseline, energy savings with RL are slightly lower than those with MPC. A much more extensive simulation study is needed to truly map out the performance boundaries of the two competing controllers. This is a topic of future research.

The slightly lower energy savings of RL observed here should be considered together with the dramatic reduction in on-line computation of the control command: RL involves solving a quadratic program with 4 decision variables, while MPC involves solving a non-convex optimization problem with 3456 decision variables (for a 24 hour long planning horizon). The low computation complexity of RL comes from its model-free nature, which also eliminates the need for identification of a control-oriented and yet accurate model, which is an important and costly aspect of MPC.

Apart from high performance, acceptance by building and equipment operators is also needed for an HVAC control technique to be adopted. RL falls under the category of artificial intelligence (AI) techniques and is not designed to be interpretable, which may hamper its adoption. The RL controller presented here is, however, interpretable to a limited extent, in the sense that the trend of its decisions—though perhaps not the numerical values—can be predicted by examining the Q-function it learns, and these decisions are also consistent with the underlying physics.

RL trades off low on-line computation cost with high off-line computation cost incurred during learning. Performance depends on the learning algorithms used, the model used for simulations during learning, amount of exploration during learning, and many design choices such as the reward function, choice of state, etc. Investigating the impact of the remaining design dimensions is an obvious avenue of future work. Extending to multi-zone building control is another topic of exploration.

References

[1] "International energy outlook 2019," Energy information administration, Department of Energy, U.S. Govt., Tech. Rep., September 2019. [Online]. Available: https://www.eia.gov/outlooks/ieo/pdf/ieo2019.pdf

[2] G. Serale, M. Fiorentini, A. Capozzoli, D. Bernardini, and A. Bemporad, "Model predictive control (MPC) for enhancing building and HVAC system energy efficiency: Problem formulation, applications and opportunities," *Energies*, vol. 11, no. 3, p. 631, 2018.

[3] P. Barooah, "Building energy management system," in *Encyclopedia of Systems and Control*, J. Baillieul and T. Samad, Eds. London: Springer London, 2019, pp. 1–7.

[4] N. Aste, M. Manfren, and G. Marenzi, "Building automation and control systems and performance optimization: A framework for analysis," *Renewable and Sustainable Energy Reviews*, vol. 75, pp. 313 – 330, 2017.

[5] E. Atam and L. Helsen, "A convex approach to a class of non-convex building HVAC control problems: Illustration by two case studies," *Energy and Buildings*, vol. 93, pp. 269 – 281, 2015.

[6] T. Zeng and P. Barooah, "An autonomous MPC scheme for energy-optimal control of building HVAC systems," in *American Control Conference*, July 2020, accepted.

[7] N. S. Raman, K. Devaprasad, and P. Barooah, "MPC-based building climate controller incorporating humidity," in *American Control Conference*, July 2019, pp. 253–260.

[8] C. J. C. H. Watkins, "Learning from delayed rewards," Ph.D. dissertation, University of Cambridge, 1989.

[9] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, 2015.

[10] K. Mason and S. Grijalva, "A review of reinforcement learning for autonomous building energy management," *arXiv.org*, 2019, arXiv:1903.05196.

[11] S. Liu and G. P. Henze, "Experimental analysis of simulated reinforcement learning control for active and passive building thermal storage inventory: Part 1. Theoretical foundation," *Energy and Buildings*, vol. 38, no. 2, pp. 142 – 147, 2006.

[12] B. Li and L. Xia, "A multi-grid reinforcement learning method for energy conservation and comfort of HVAC in buildings," in *2015 IEEE International Conference on Automation Science and Engineering (CASE)*. IEEE, 2015, pp. 444–449.

[13] Z. Zhang and A. Chong, "A deep reinforcement learning approach to using whole building energy model for HVAC optimal control," in *Building Performance Modeling Conference and SimBuild*, 2018.

[14] T. Wei, Y. Wang, and Q. Zhu, "Deep reinforcement learning for building HVAC control," in *Proceedings of the 54th Annual Design Automation Conference 2017*, ser. DAC '17. New York, NY, USA: ACM, 2017, pp. 22:1–22:6.

[15] T. Wei, S. Ren, and Q. Zhu, "Deep reinforcement learning for joint datacenter and HVAC load control in distributed mixed-use buildings," *IEEE Transactions on Sustainable Computing*, 2019.

[16] C. Zhang, Z. Zhang, and V. Loftness, "Bio-sensing and reinforcement learning approaches for occupant-centric control," *ASHRAE Transactions*, vol. 125, pp. 374–381, 2019.

[17] A. M. Devraj and S. Meyn, "Zap Q-learning," in *Advances in Neural Information Processing Systems*, 2017, pp. 2235–2244.

[18] S. Chen, A. M. Devraj, A. Bušić, and S. Meyn, "Zap Q-learning with nonlinear function approximation," *arXiv.org*, 2019, arXiv:1910.05405.

[19] ASHRAE, "The ASHRAE handbook : Applications (SI Edition)," 2011.

[20] N. S. Raman, A. M. Devraj, P. Barooah, and S. P. Meyn, "Reinforcement learning for control of building HVAC systems," in *American Control Conference*, July 2020, accepted.

[21] N. S. Raman and P. Barooah, "On the round-trip efficiency of an HVAC-based virtual battery," *IEEE Transactions on Smart Grid*, vol. 11, no. 1, pp. 403–410, Jan 2020.

[22] K. J. Astrom and R. M. Murray, *Feedback Systems: An Introduction for Scientists and Engineers*. USA: Princeton University Press, 2008.

[23] R. Sutton and A. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. Cambridge, MA: MIT Press. On-line edition at http://www.cs.ualberta.ca/~sutton/book/the-book.html, 2018.

[24] ASHRAE, "ANSI/ASHRAE standard 62.1-2016, ventilation for acceptable air quality," 2016.

[25] D. Choi and B. Van Roy, "A generalized Kalman filter for fixed point approximation and efficient temporal-difference learning," *Discrete Event Dynamic Systems*, vol. 16, no. 2, pp. 207–239, 2006.

[26] S. Shivam, I. Buckley, Y. Wardi, C. Seatzu, and M. Egerstedt, "Tracking control by the Newton-Raphson flow: Applications to autonomous vehicles," *arXiv.org*, 2018, arXiv:1811.08033.

[27] Y. Wardi, C. Seatzu, M. Egerstedt, and I. Buckley, "Performance regulation and tracking via lookahead simulation: Preliminary results and validation," in *Conference on Decision and Control*, December 2017, pp. 6462–6468.

[28] A. M. Devraj and S. P. Meyn, "Fastest convergence for Q-learning," *arXiv.org*, 2017, arXiv:1707.03770.

[29] N. Raman, K. Devaprasad, B. Chen, H. A. Ingley, and P. Barooah, "MPC for energy efficient HVAC control with humidity and latent heat considerations," *arXiv.org*, 2019, arXiv:1903.04652.

[30] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi: A software framework for nonlinear optimization and optimal control," *Mathematical Programming Computation*, Jul 2018, , first available online.

[31] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical Programming*, vol. 106, no. 1, pp. 25–57, Mar 2006.