Deep Reinforcement Learning for Building HVAC Control

Tianshu Wei
University of California, Riverside
twei002@ucr.edu

Yanzhi Wang
Syracuse University
ywang393@syr.edu

Qi Zhu
University of California, Riverside
qzhu@ece.ucr.edu

ABSTRACT

Buildings account for nearly 40% of the total energy consumption in the United States, about half of which is used by the HVAC (heating, ventilation, and air conditioning) system. Intelligent scheduling of building HVAC systems has the potential to significantly reduce the energy cost. However, the traditional rule-based and model-based strategies are often inefficient in practice, due to the complexity in building thermal dynamics and heterogeneous environment disturbances. In this work, we develop a data-driven approach that leverages the deep reinforcement learning (DRL) technique, to intelligently learn the effective strategy for operating the building HVAC systems. We evaluate the performance of our DRL algorithm through simulations using the widely-adopted EnergyPlus tool. Experiments demonstrate that our DRL-based algorithm is more effective in energy cost reduction compared with the traditional rule-based approach, while maintaining the room temperature within desired range.

1. INTRODUCTION

Buildings account for nearly 40% of the total energy consumption in the United States and 70% of the electricity usage [23]. Among various types of building energy loads, such as HVAC, lighting, appliances and electric vehicle charging, the HVAC system consumes around 50% of the energy usage. The thermal flywheel effect allows the building operator to perform pre-cooling/per-heating to *shift* the HVAC energy demand while still meeting the requirements on room temperature [16]. As the energy usage in buildings is often charged with time-of-use price (where energy prices vary in different time periods), the scheduling flexibility from HVAC system provides great potential for reducing building energy cost and improving grid energy efficiency and stability.

In the literature, many approaches have been proposed to control building HVAC systems for energy efficiency [15, 11, 10, 25]. These approaches typically employ a simplified building thermal dynamics model during runtime control to predict buildings' temperature evolution. In [10], the authors develop a nonlinear model of the overall cooling system, including chillers, cooling towers and thermal storage banks, and present a model predictive control (MPC) scheme for minimizing energy consumption. In [15], the authors propose a system model that is bilinear in inputs, states and weather parameters, and model the control optimization as a sequential linear programming (SLP) problem. The work in [11] models building thermal behavior as RC networks and proposes a tracking linear-quadratic regulator (LQR) for HVAC control. The work in [25] uses the similar building model as [11], and develops an MPC-based algorithm for co-scheduling HVAC control with other demands and supplies. The performance and reliability of these approaches depend highly on the accuracy of the building thermal dynamics model, which also has to be efficiently solved using mathematical tools for practical runtime control [27]. However, the building temperature is affected by many factors, including building structure and materials, surrounding environment (e.g., ambient temperature, humidity, and solar radiation intensity), and internal heat gains from occupants, lighting systems and other equipment. As a result, the building temperature often exhibits randomized behaviors under an incomplete modeling. *Overall, it is often intractable to develop a building dynamics model that is both accurate and efficient enough for effective runtime HVAC control.*

Therefore, some recent works have started developing *data-driven* approaches that leverage real-time data inputs for reinforcement learning (RL) based HVAC control. In [1, 9, 14], classical Q-learning techniques are presented, which use the tabular Q value function and are not suitable for control problems with large state space. In [6], the authors propose a neural fitted RL method through the interaction with tenants to determine the optimal temperature setting point. This approach is only evaluated with single-zone buildings, in which the heat transfer process is modeled by simple form of differential equations. The work in [3] develops a model-assisted batch RL approach, where randomized trees are used to approximate the action value and decide simple on-off control strategies. These RL-based methods with function approximation work for the continuous state situation. However, their batch update mechanism exhibits high computational cost throughout the learning process.

The recently proposed *deep reinforcement learning* (DRL) technique, which has been shown successful in playing Atari and Go games [12, 19], emerges as a powerful data-driven method for solving complex control problems. The DRL technique can handle large state space by building a deep neural network to relate the value estimates and associated state-action pairs, thereby overcoming the shortcoming of conventional RL. This paper is the first (to the best of authors' knowledge) to apply DRL technique for HVAC control. Our work achieves good performance and high scalability by (1) formulating the HVAC operation as a Markov decision process (MDP) [1], (2) developing a DRL-based control

DAC '17, June 18-22, 2017, Austin, TX, USA
© 2017 ACM. ISBN 978-1-4503-4927-7/17/06…$15.00
DOI: http://dx.doi.org/10.1145/3061639.3062224

[1] Our MDP formulation is general and can be time-variant as well.

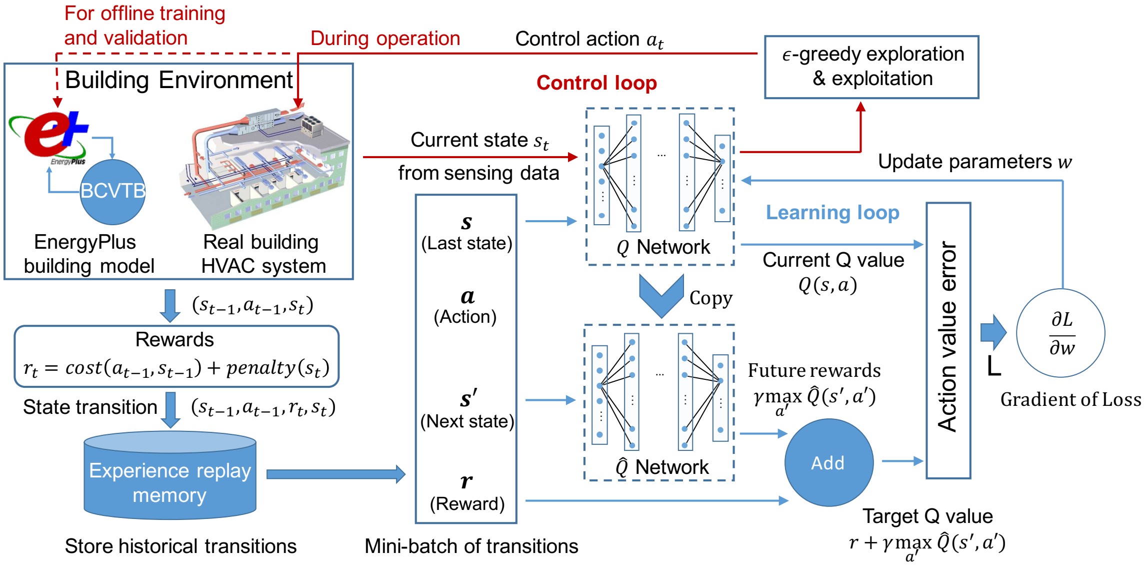

For offline training and validation
During operation
Control action a_t
ϵ-greedy exploration & exploitation
Building Environment
Control loop
BCVTB
EnergyPlus building model
Real building HVAC system
Current state s_t from sensing data
Update parameters w
Learning loop
s (Last state)
a (Action)
s' (Next state)
r (Reward)
Q Network
Copy
$\hat{Q}$ Network
Current Q value $Q(s,a)$
Future rewards $\gamma \max_{a'} \hat{Q}(s',a')$
Add
Action value error
L
$\frac{\partial L}{\partial w}$
Gradient of Loss
(s_{t-1}, a_{t-1}, s_t)
Rewards $r_t = cost(a_{t-1}, s_{t-1}) + penalty(s_t)$
State transition $(s_{t-1}, a_{t-1}, r_t, s_t)$
Experience replay memory
Store historical transitions
Mini-batch of transitions
Target Q value $r + \gamma \max_{a'} \hat{Q}(s',a')$

Figure 1: Our deep reinforcement learning (DRL) based framework for HVAC control and evaluation. The details of building state transition are defined in Section 2. The details of DRL learning and control process are presented in Section 3.

framework and an efficient heuristic variant, and (3) facilitating algorithm training and evaluation with a co-simulation framework. Figure 1 illustrates our DRL-based framework for HVAC control and evaluation. During building operation, it learns an effective control policy based on sensing data input, without relying on any thermal dynamics model. For offline training and validation of the algorithm, we leverage detailed building dynamics model built in the widely-adopted *EnergyPlus* simulation tool [4]. Simulation results demonstrate that the proposed framework is able to significantly reduce the energy cost while meeting the room temperature requirements. It should be noted that while the detailed EnergyPlus models are highly accurate and suitable for offline training and validation, their high complexity makes them unsuitable for real-time control.

In summary, the *main contributions* of our paper include:

- We formulate HVAC control operations as a Markov decision process, with definitions of system state, control action and reward function. These are the key concepts in our data-driven HVAC control approach using DRL.
- We develop a DRL based algorithm for minimizing the building energy cost while maintaining comfortable temperature for building tenants. For higher scalability, we further propose a heuristic variant for efficient control for complex multiple-zone systems.
- We develop a co-simulation framework based on EnergyPlus for offline training and validation of our DRL-based algorithms, with real-world weather and time-of-use pricing data. Our experiment results demonstrate that the proposed DRL-based algorithms can achieve up to $20\% - 70\%$ energy cost reduction when compared with a rule-based baseline control strategy.

The remainder of the paper is organized as follows. Section 2 presents our MDP formulation for the HVAC control operation. Section 3 presents our DRL-based HVAC control algorithms. Section 4 shows the experimental results and Section 5 concludes the paper.

2. MDP FORMULATION FOR DRL-BASED BUILDING HVAC CONTROL

The building HVAC system is operated to maintain a desired temperature within each zone, based on current temperature and outside environment disturbances. The zone temperature at next time step is only determined by the current system state and environment disturbances, and the conditioned air input from the HVAC system. It is independent from the previous states of the building. Therefore, the HVAC control operation can be treated as a Markov decision process. Next, we formulate the key concepts in this process to facilitate our DRL-based HVAC control algorithm.

Control actions: We consider a building that has z temperature zones and is equipped with a VAV (variable air flow volume) HVAC system. The VAV terminal box at each zone provides conditioned air (typically at a constant temperature) with an air flow rate that can be chosen from multiple discrete levels, denoted as $F = \{f^1, f^2, ..., f^m\}$. Therefore, the entire action space $A = \{\mathcal{A}^1, \mathcal{A}^2, ..., \mathcal{A}^n\}$ of the building HVAC control includes all possible combinations of air flow rate for every zone, i.e., $n = m^z$. Clearly, the dimension of action space will increase rapidly with larger number of zones and air flow rate levels, which will then greatly increase the training time and degrade the control performance. In Section 3.3, we introduce a multi-level control heuristic for multiple zones to combat this challenge.

System states: The optimal control action is determined based on the observation of the current system state. In this work, we consider current (physical) time, zone temperature and environment disturbances (i.e. ambient temperature and solar irradiance intensity) to determine the optimal control action. In particular, incorporating current time information in the state enables the DRL algorithm to adapt to time related activities, such as time-varying temperature requirements, electricity price, occupant activities and equipment operation in the building. For environment disturbances, instead of just using current ambient temperature and solar irradiance, we also take into account of multi-step forecast of weather data. This is important because the weather pattern can vary significantly. Considering a short sequence of weather forecast data enables our DRL algorithm to capture the trend of the environment, perform proactive control and adapt to time-variant systems.

Rewards function: The goal of the DRL algorithm is to minimize the total energy cost while maintaining the temperature of each zone within a desired range, by taking a sequence of actions $\{a_1, a_2, \dots, a_t\}$, where $a_t \in A$. After taking an action a_{t-1} at state s_{t-1}, the building will evolve into a new state s_t and the DRL algorithm will receive an immediate reward r_t, as calculated below in Equation (1).

$$r_t = -cost(a_{t-1}, s_{t-1}) - \lambda \sum_{i=1}^{z}([T_t^i - \overline{T}_t^i]_+ + [\underline{T}_t^i - T_t^i]_+) \quad (1)$$

which includes the energy cost of the last control action a_{t-1} and the total penalty of temperature violation. We use negative rewards as our DRL algorithm will maximize the total reward. It should be noted that the goal of minimizing energy cost contradicts the goal of maintaining desired temperature, and the reward function tries to balance the two.

During the operation of HVAC systems, we want to maximize the accumulative reward $R = \sum_{i=1}^{\infty} \gamma^{i-1} r_{t+i}$, where $\gamma \in [0, 1]$ is a *decay factor* that controls the window length when maximizing the reward. We use $Q^*(s_t, a_t)$, i.e., the *optimal value*, to represent the maximum accumulative reward we can obtain by taking action a_t in state s_t. $Q^*(s_t, a_t)$ can be calculated by Bellman Equation (2) in a recursive fashion.

$$Q^*(s_t, a_t) := E[r_{t+1} + \gamma \max_{a_{t+1}} Q^*(s_{t+1}, a_{t+1}) | s_t, a_t] \quad (2)$$

The state transition in buildings is stochastic, because the zone temperature is affected by various disturbances, which cannot be accurately measured. In this work, we update the *value estimates* by following the Q-learning [24] method, as shown in Equation (3).

$$\begin{aligned} Q_{t+1}(s_t, a_t) &:= Q_t(s_t, a_t) \\ &+ \eta(r_{t+1} + \gamma \max_{a_{t+1}} Q_t(s_{t+1}, a_{t+1}) - Q_t(s_t, a_t)) \quad (3) \end{aligned}$$

where $\eta \in (0, 1]$ represents the learning rate of value estimates during the training process. Equation (3) should converge to the optimal value $Q^*(s_t, a_t)$ over time under the MDP environment.

Building control sequence: Our DRL algorithm interacts with the building environment during operation, or with the EnergyPlus model via the *BCVTB* (a Ptolemy II platform that enables co-simulation across different models [26]) interface during offline training and validation.

As shown in Figure 2, we use a separate control step $\Delta t_c = k\Delta t_s$ to represent the control frequency of DRL algorithm. Every Δt_c time, as shown in Equation (4), the DRL algorithm will observe the building state and update the control action. Between two control time steps, the control action used to operate the HVAC system remains the same as the last updated action. While in Equation (5), the building receives the control signal and enters its next state every Δt_s time, which represents the building simulation or sensor sampling frequency.

$$a_t = f_{DRL}(s_{t-\Delta t_s}) \quad (4)$$

$$s_t = f_{ENV}(s_{t-\Delta t_s}, a_{t-\Delta t_s}) \quad (5)$$

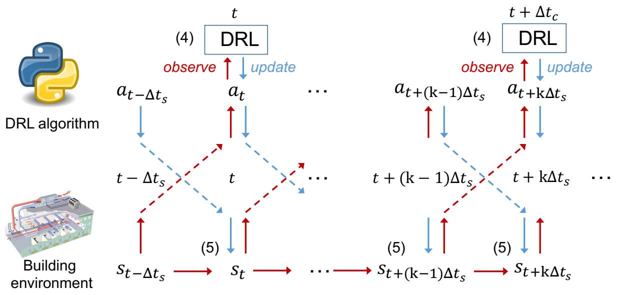


Figure 2: Building control sequence with DRL algorithm

3. DRL-BASED HVAC CONTROL

3.1 Value Function Approximation

The combination of possible values of each feature in the state vector forms a very large state space. In practice, it is more efficient to use generalization methods, such as randomized trees [5], kernel-based method [17] and neural networks [18] to approximate the Q-value. In this work, we use the artificial neural network to approximate the Q-value calculated by Equation (3). As shown in Figure 3, we adopt a similar neural network structure as in [12]. With this structure, the Q-value estimates for all control actions can be calculated by performing one forward pass (inference) in the neural network. This can greatly improve the efficiency when selecting actions with the ϵ-greedy policy. The input features of the network are the environment state that is defined in Section 2. The *rectified linear unit* (ReLU) is used as the activation function for hidden layers, and the linear layer is used for inferring action value at the output.

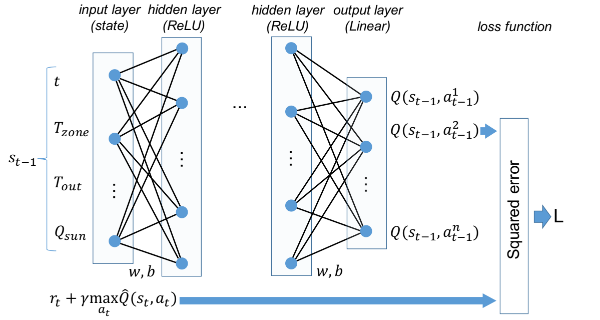


Figure 3: Structure of the neural network utilized in our DRL framework

We use the mean squared error between the target Q-value and the inferred output of neural network as loss function (6), where n denotes the number of possible control actions. Parameters (weights) in the neural network are updated by the mini-batch gradient descent method $w := w - \alpha \Delta w$ [2], where α is the learning rate and $\Delta w = \frac{\partial L}{\partial w}$.

$$L = \frac{1}{2n} \sum_{i=1}^{n} [Q^*(s_t, a_t^i) - Q(s_t, a_t^i)]^2 \quad (6)$$

Being consistent with the Q-learning update process (3), the target value $Q^*(s_t, a_t^i)$ in the neural network can be estimated by Equation (7) when using gradient descent, where Q values are approximated by the neural network.

$$Q^*(s_t, a_t) = r_{t+1} + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}) \quad (7)$$

Training data pre-processing: The input state vector s_t consists of various types of features in the building. The range of value for each feature can vary significantly. To facilitate the learning process, we scale the feature values to a similar range before feeding the input state to the neural network. In this work, we scale the input state vector to the range $[0, 1]$ as shown in (8), where x represents a feature in the input state. The minimum and maximum values for each feature can be estimated from historical observations.

$$x' = \frac{x - min(x)}{max(x) - min(x)} \tag{8}$$

For output units, the linear layer is used to infer Q-value estimates from hidden units. However, if we directly use reward function (1) to calculate the target Q-value as shown in (7), it may result in a large variance in the target value. During backward propagation, the corresponding bias factor in the last linear layer may dominate the derivative of the loss function, which will prevent weights in earlier layers from learning the optimal value. In order to overcome this limitation, we calculate the target value by first shrinking the original immediate reward with a factor ρ and then clipping it if the target is smaller than -1, as shown in Equation (9).

$$target\_val(s_{t-1}, a_{t-1}) = \max[\frac{r_t}{\rho} + \gamma \max_{a_t} Q(s_t, a_t), -1] \tag{9}$$

In this way, we squash the original target value with a large variance to the range $[-1, 0]$. *The underlying principle is that while it does not help to know which control actions are worse, we focus on which control actions are better.*

Training of the neural network: As shown in Figure 1, the one-step state transition process is represented by a tuple $(s_{t-1}, a_{t-1}, r_t, s_t)$, which includes previous state, previous action, immediate reward and current state. The *target vector* of the neural network can be calculated by Equation (10), where the target value associated with a_{t-1}, i.e., $target\_val(s_{t-1}, a_{t-1})$, is calculated by Equation (9). For other control actions, the target value is set to the current value estimate of that action.

$$target(s_{t-1}) = \begin{cases} target\_val(s_{t-1}, a) & \text{if } a = a_{t-1} \\ Q(s_{t-1}, a) & \text{otherwise} \end{cases} \tag{10}$$

Next, the target vector is compared with current inference output of the neural network to calculate the approximation error. Then, we use the RMSprop [8] method to update parameters in the neural network.

3.2 DRL Algorithm Design

Our DRL-based HVAC control algorithm is presented in Algorithm 1. The outer loop controls the number of training episodes, while the inner loop performs HVAC control at each simulation time step within one training episode.

Initial setup: During the learning process, the recent transition tuples $(s_{t-1}, a_{t-1}, r_t, s_t)$ are stored in the memory M, from which a mini-batch of samples will be generated for neural network training. At the beginning, we first initialize memory M as an empty set. Then, we initialize weights w in the neural network similar as in [7]. As shown in Equation (7), updating neural network weights requires the target value, which also depends on weights in the neural network. To break this dependency loop between target value and weights w, in line 3, a separate neural network $\hat{Q}$ is created for calculating the target value similar as in [12]. This network $\hat{Q}$ will be periodically updated by copying parameters from the network Q during the learning process. In line 4, the variable a stores the control action in the last step, and s_{pre} and s_{cur} represent the building state in the previous and current control time steps, respectively.

Algorithm 1 DRL-based HVAC Control Algorithm

```
 1: Initialize memory $M$ = [empty set]
 2: Initialize neural network $Q$ with parameters $w$
 3: Copy neural network $Q$ and store as $\hat{Q}(\cdot|\hat{w})$
 4: Initialize control action $a$, state $s_{pre}$ and $s_{cur}$
 5: for $m := 1$ to $N$ do
 6:     Reset building environment to initial state
 7:     for $t_s := 0$ to $L$ do
 8:         if $t_s \ mod \ k \ == \ 0$ then
 9:             $s_{cur} \leftarrow current \ observation$
10:             $r = reward(s_{pre}, a, s_{cur})$
11:             $M \leftarrow (s_{pre}, a, r, s_{cur})$
12:             Draw mini-batch $(s, a, r, s') \leftarrow M$
13:             Target vectors $v \leftarrow target(s)$
14:             Train $Q(\cdot|w)$ with $s, v$
15:             Every $d\Delta t_c$ steps, $\hat{Q}(\cdot|\hat{w}) \leftarrow Q(\cdot|w)$
16:             $\epsilon = \max(\epsilon - \Delta\epsilon, \epsilon_{min})$
17:             $a = \begin{cases} \mathcal{A}^i \in A \mid i = random(n) & \text{probability } \epsilon \\ \underset{\tilde{a}}{argmax} Q(s_{cur}, \tilde{a}) & \text{otherwise} \end{cases}$
18:             $s_{pre} \leftarrow s_{cur}$
19:         end if
20:         Execute action $a$ in building environment
21:     end for
22: end for
```

Learning process: Within each training episode, line 8 determines whether the current time step t_s is a control time step. As discussed in Section 2, the control step Δt_c is k times of the simulation step Δt_s. If t_s is a control time step, the algorithm will perform training and determine the new control action (line 9 to 18). Otherwise, the building will maintain the current control action.

During the learning process, in line 9 we first observe the state at current control time step. Then, the immediate reward is calculated by Equation (1). Next, in line 11 the state transition tuple is stored in memory. Then, a mini-batch of transition tuples are drawn randomly from the memory. Lines 13 to 14 follow Equation (10) to calculate the target vector and update weights in neural network Q by using the *RMSprop* Back-propagation method [8]. In line 15, the network $\hat{Q}$ will be updated with current weights in network Q in every d control time steps. Then, this $\hat{Q}$ network is used for inferring the target value for the next d control steps.

Next, from line 16 to 18 the network Q is utilized to determine the next control action. The ϵ-greedy policy is used to select the optimal control action based on the output of Q. The algorithm has a probability ϵ to explore the action space by randomly selecting an available action; otherwise, it will choose the action with the maximum value estimate. After each training process, in line 16 the exploration rate ϵ will gradually decrease until reaching at a lower bound ϵ_{min}. In this way, the DRL algorithm is more likely to try different control actions at the beginning. As the training process proceeds, the DRL algorithm will have a higher chance to follow the learned policy. Finally, in line 18 the current state is assigned to s_{pre} to prepare for the next training process.

3.3 Heuristic Adaption for Multiple Zones

We present a heuristic mechanism that adapts our DRL algorithm for multi-zone HVAC control. As discussed in Section 2, the action space has a cardinality of m^z, which increases exponentially with the number of zones in the building. Training a neural network with such a large number of outputs is inefficient or even infeasible in practice.

In our heuristic, instead of using a single neural network to approximate the Q-values of all control actions in the building, we separately train a neural network for each zone using Algorithm 1. Each neural network is responsible for approximating the Q-value in one zone. At each time step, all networks will receive the state of buildings, and then determine the control action for each zone separately. After executing the control action, the temperature violation penalty for each zone is calculated similarly as Equation (1). The electricity cost in each zone is calculated by Equation (11), which is proportional to the air flow demand in each zone based on the total cost.

$$cost_i = cost \cdot \frac{u_i}{\sum_i u_i} \tag{11}$$

where *cost* denotes the total electricity cost in the building and u_i represents the air flow rate in each zone. Although the total electricity cost is not exactly a linear function of the air flow rate, we can still heuristically estimate the amount of cost contributed by each zone by following Equation (11).

4. EXPERIMENTAL RESULTS

4.1 Experiment Setup

We demonstrate the effectiveness of our DRL-based algorithms through simulations in EnergyPlus. We train the DRL algorithms on weather profiles of summer days in two areas, obtained from the National Solar Radiation Data Base [13]. The weather data from Area 1 (Riverside) has intensive solar radiation and large variance in temperature, while Area 2 (Los Angeles) has a milder weather profile. We use the practical time-of-use price from the Southern California Edison [20] to calculate buildings' electricity cost. The desired temperature range is between $19°C$ and $24°C$ based on the ASHRAE standard [21]. There are 4 hidden layers in the neural network. The network layout and other parameters in our DRL algorithms are listed in Table 1. We train our DRL algorithm using 100 episodes (months) of data. In practice, the training process can be facilitated by building accurate EnergyPlus models.

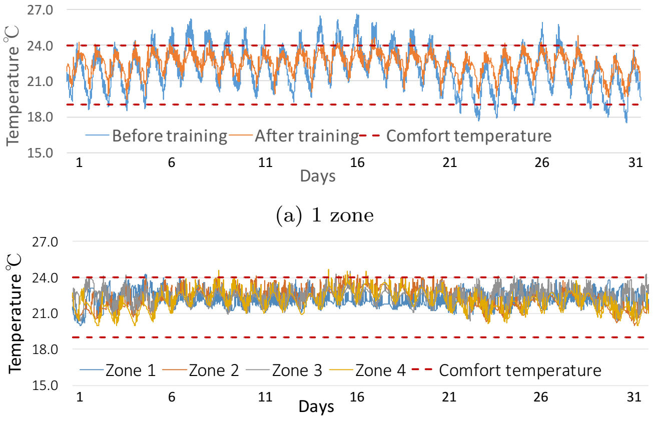
Table 1: Parameter settings in DRL Algorithms

Δt_s	$1\ min$	Δt_c	$15\ min$
k	15	d	$48 * 5$
mini-batch	48	memory size	$48 * 31$
η	0.99	α	0.003
ρ	1000	λ	100
ϵ_{min}	0.1	N	100
$\underline{T}$	$19°C$	$\overline{T}$	$24°C$
number of neurons		50, 100, 200, 400	

We evaluate the performance of our DRL algorithms by comparing them with a rule-based HVAC control strategy (similarly as the one in [22]) and the conventional RL method. In the rule-based approach, the HVAC system is operated by an on-off control strategy such that if the zone temperature exceeds the cooling setpoint (i.e. $24°C$ in our experiment), the room will be cooled at the maximum air flow rate. If the temperature drops below certain threshold [2], the air flow in the zone will be turned off. In our experiment, for both baseline approaches and DRL algorithms, the conditioned air temperature from the HVAC system is set to $10°C$.

4.2 Experiment Results

A. Effectiveness of DRL control algorithms in meeting temperature requirements: We evaluate the performance of our DRL algorithms with three buildings modeled in EnergyPlus, which have 1 zone, 4 zones and 5 zones, respectively. The HVAC system can provide multi-level air flow rate for each zone. In this work, we test our DRL algorithms with two-level (i.e. on-off control) and five-level air flow control, where each level is evenly distributed between the minimum and maximum air flow rate of the HVAC system. Figure 4 shows the zone temperature in the 1-zone and 4-zone building in August, where our regular DRL algorithm in Algorithm 1 performs on-off control to operate the HVAC system. We can see that after training the DRL algorithm is quite effective in maintaining the zone temperature within the desired range.

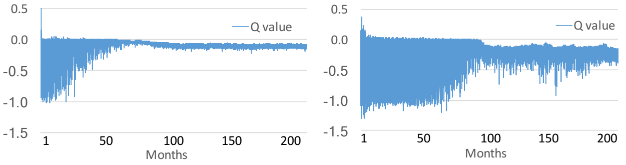


(a) 1 zone

(b) 4 zones

Figure 4: Effectiveness of our regular DRL algorithm in maintaining comfort temperature

Figure 5 shows the average Q value of our regular DRL algorithm throughout the learning process. At the beginning, the Q value is very small due to the large penalty caused by frequent temperature violations. The Q value will gradually increase as the DRL algorithm learns the effective strategy to maintain the zone temperature within the desired range. Eventually, the Q value will stabilize when the DRL algorithm learns the policy to avoid temperature violation and minimize the electricity cost.



(a) 1 zone

(b) 4 zones

Figure 5: Q value in 1-zone and 4-zone buildings

[2] We find out that setting it to $20°C$ helps the rule-based approach minimize temperature violation rate.

As discussed in Section 3.3, the action space will exponentially increase with the number of zones. For the 5-zone building, the total number of actions is more than 3000 with 5-level air flow rate control, which would be intractable for our regular DRL algorithm. Therefore, we leverage the efficient heuristic method in Section 3.3 to perform multi-level control in multi-zone buildings. Figure 6 compares the average frequency of temperature violations of the baseline strategy, conventional Q learning, our regular DRL algorithm with on-off control, and the heuristic DRL algorithm with 5-level control. *We can see that the DRL algorithms are able to keep the percentage of temperature violations at a low level.*

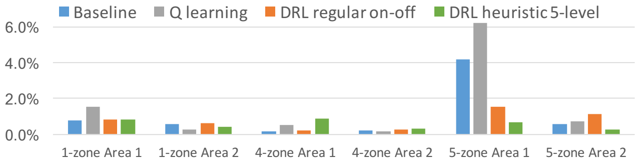


Figure 6: Comparison of temperature violation rate between our DRL algorithm, baseline approach and Q learning

B. Effectiveness of DRL algorithms in energy cost reduction: Figure 7 shows the comparison of average daily electricity cost of our DRL control algorithms, conventional Q learning and the baseline approach. The percentage of cost reduction achieved by DRL algorithms compared with the baseline approach is marked in the figure.

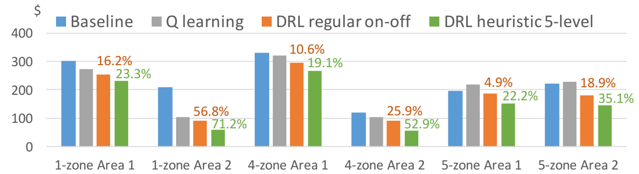


Figure 7: Comparison of energy cost between our DRL algorithms, baseline approach and Q learning

We can see that our regular DRL algorithm can achieve significant energy cost reduction compared with the baseline approach and conventional Q learning. The efficient heuristic DRL can leverage multi-level control in multi-zone buildings to achieve further reduction. Furthermore, the DRL algorithms are more effective in reducing energy cost for Area 2, since the learning process is more effective with a milder weather profile. Compared with the 5-zone building, the DRL algorithms achieve more reduction for 1-zone and 4-zone buildings. That is likely because the 5-zone building is more sensitive to outside disturbances and hence more challenging for the learning process.

5. CONCLUSIONS

This paper presents a deep reinforcement learning based data-driven approach to control building HVAC systems. A co-simulation framework based on EnergyPlus is developed for offline training and validation of the DRL-based approach. Experiments with detailed EnergyPlus models and real weather and pricing data demonstrate that the DRL-based algorithms (including the regular DRL algorithm and a heuristic adaptation for efficient multi-zone control) are able to significantly reduce energy cost while maintaining the room temperature within desired range.

Acknowledgments. The authors gratefully acknowledge the support from the National Science Foundation award CCF-1553757 and the Riverside Public Utilities.

6. REFERENCES

[1] E. Barrett and S. Linder. *Autonomous HVAC Control, A Reinforcement Learning Approach*. Springer, 2015.

[2] L. Bottou. Large-scale machine learning with stochastic gradient descent. *Proceedings of COMPSTAT*. 2010.

[3] G. T. Costanzo and et al. Experimental analysis of data-driven control for a building heating system. *CoRR*, abs/1507.03638, 2015.

[4] EnergyPlus. https://energyplus.net/.

[5] D. Ernst and et al. Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research*, 2005.

[6] P. Fazenda and et al. Using reinforcement learning to optimize occupant comfort and energy usage in hvac systems. *Journal of Ambient Intelligence and Smart Environments*, pages 675–690, 2014.

[7] K. He and et al. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *IEEE International Conference on Computer Vision*, 2015.

[8] G. Hinton, N. Srivastava, and K. Swersky. Lecture 6a overview of mini–batch gradient descent. http://www.cs.toronto.edu/˜tijmen/csc321/slides/lecture_slides_lec6.pdf.

[9] B. Li and L. Xia. A multi-grid reinforcement learning method for energy conservation and comfort of HVAC in buildings. pages 444–449, 2015.

[10] Y. Ma and et al. Model predictive control for the operation of building cooling systems. *IEEE Transactions on Control Systems Technology*, 20(3):796–803, 2012.

[11] M. Maasoumy and et al. Model-based hierarchical optimal control design for HVAC systems. *DSCC*, 2011.

[12] V. Mnih and et al. Human-level control through deep reinforcement learning. *Nature 518.7540*, 2015.

[13] National Solar Radiation Data Base. http://rredc.nrel.gov.

[14] D. Nikovski, J. Xu, and M. Nonaka. A method for computing optimal set-point schedules for HVAC systems. *REHVA World Congress CLIMA*, 2013.

[15] F. Oldewurtel and et al. Energy efficient building climate control using stochastic model predictive control and weather predictions. *ACC*, 2010.

[16] S. J. Olivieri and et al. Evaluation of commercial building demand response potential using optimal short-term curtailment of heating, ventilation, and air-conditioning loads. *Journal of Building Performance Simulation*, 2014.

[17] D. Ormoneit and Ś. Sen. Kernel-based reinforcement learning. *Machine Learning*, 49(2):161–178, 2002.

[18] M. Riedmiller. *Neural Fitted Q Iteration – First Experiences with a Data Efficient Neural Reinforcement Learning Method*. Springer, 2005.

[19] D. Silver and et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587), 2016.

[20] SCE. https://www.sce.com/NR/sc3/tm2/pdf/CE281.pdf.

[21] A. Standard. Standard 55-2004-thermal environmental conditions for human occupancy. *ASHRAE Inc.*, 2004.

[22] D. Urieli and P. Stone. A learning agent for heat-pump thermostat control. *AAMAS*, 2013.

[23] U.S. DoE. Buildings energy data book.

[24] C. J. Watkins and P. Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

[25] T. Wei, Q. Zhu, and M. Maasoumy. Co-scheduling of HVAC control, EV charging and battery usage for building energy efficiency. *ICCAD*, 2014.

[26] M. Wetter. Co-simulation of building energy and control systems with the building controls virtual test bed. *Journal of Building Performance Simulation*, 2011.

[27] L. Yang and et al. Reinforcement learning for optimal control of low exergy buildings. *Applied Energy*, 2015.