A Simplified Building Controls Environment with a Reinforcement Learning Application

Vasken Dermardiros[1], Scott Bucking[2], Andreas K. Athienitis[1]
[1] Centre for Zero Energy Building Studies, Department of Building, Civil and Environmental Engineering, Concordia University, Canada
[2] Department of Civil and Environmental Engineering, Carleton University, Canada

Abstract

This paper has two contributions: (1) it describes the design and implementation of a thermal network based building zone emulator environment suitable for control applications with inputs and outputs matching those of the OpenAI Gym environments; and (2) the use of the environment to train a proximal policy optimization (PPO) reinforcement learning (RL) agent to maintain comfort for a house while minimizing the HVAC system energy consumption. The building controls environment can be extended to many other cases including training agents that can be mass-applied, studying the behaviour of communities and studying occupant behaviour and comfort. The trained RL agent was compared to conventional bang-bang and proportional-integral controllers. Given the penalty weights, the RL agent was more adept at minimizing equipment cycling thus improving equipment longevity and efficiency.

Introduction

As more decentralized renewable energy sources are introduced onto the grid, utility level load balancing is becoming an issue. Photovoltaics can potentially create local generation peaks when the loads are low and thus the grid may not be able to absorb this production. Wind turbines rely on wind which fluctuates. Recently, Google DeepMind is in talks with UK's National Grid to help balance their electricity demand due to their larger reliance on energy from intermittent renewable sources (Thomas, 2017).

Regulation can occur on the utility side through modulating power plants or using large scale storage solutions, or on the consumption side through incentives and direct links with intelligent thermostats. Intelligent buildings and homes can aid the utility by relying on energy flexibility concepts such as optimally controlling their thermal and/or electrical storage systems (Denholm and Hand, 2011; Jensen et al., 2017; Morales et al., 2014).

Nest (Nhede, 2018) reports being able to save its US customers 12% in heating and 15% in cooling costs by shifting the usage period from peak times to when energy demand is lower and cheaper. Ecobee (Ecobee, 2018) another smart thermostat manufacturer, has a "donate your data" program where customers can share anonymized data to help researchers develop more energy efficient control strategies by having a statistical representation of the population. Even with an added layer of *smartness*, the underlying control algorithms in these thermostats are conventional and reactive. They condition the space based on a setpoint error: amount of heating or cooling is proportional to a difference in a desired and measured temperatures, with a deadband to reduce excessive cycling of the equipment. There is seldom reliance on the use of weather or occupancy predictions and on the use of the building characteristics itself. There is work on applying model-based predictive controls (MPC) on buildings, but it requires an accurate model of the system to be controlled which can be cumbersome to build (Shaikh et al., 2014).

Artificial intelligence (AI) techniques can be used to learn a surrogate model of the building using a black-box machine learning (ML) approach (Gaussian process regression models, support vector regression, random forests, neural network models) or using a grey-box Bayesian regression method (applied to a generalized linear model or other) where the learned parameters are physically interpretable. These learned models would then be used in a model-based predictive control (MPC) framework to control the system by minimizing its cost function. Alternatively, the model can be learned *internally* by a reinforcement learning (RL) agent which then uses it to apply actions to maximize its rewards. The learned model remains within the agent's *brain* and is not accessible by other external programs. The RL case is presented in this paper.

RL has received tremendous attention lately after having defeated the world's strongest Go player trained purely by self-play (Silver et al., 2017) and performing at a professional level in an imperfect information game of Dota 2 (OpenAI, 2018), and more recently, beat professional players in the real-time-strategy game StarCraft II[1]. Contrary to IBM's Deep Blue chess program, these methods are not domain specific and the training methods can be generally applied to any problem that can be formulated as a Markov decision process – with care, this assumption can be relaxed. In a nutshell, an RL agent interacts with its environment to come up with a policy – what action to take when in a certain state – to maximize its sum of future discounted rewards. There exist many learning schemes and high-quality implementations are made available online through open source libraries such as OpenAI Baselines (Dhariwal et al., 2017) and Tensorforce (Schaarschmidt et al., 2017). These libraries plug their agents into standardized environments like those found in OpenAI Gym. By creating an environment that matches these requirements, we can directly leverage cutting-edge advances in RL into our building engineering and controls domain by using these libraries instead of having to re-implement the logic.

To briefly overview the RL process, an *agent* is placed in an *environment* where it observes the *state* it's in and decides on what *action* to take based on its *policy*. Given the *action*, the *environment* responds and returns an updated *state* and, optionally, a *reward*, see Figure 1. This process continues until the end of the *episode*, after which, the *environment* is reset and the process is ran again. Every episode is independent of one another. The agent's task is to apply actions as to maximize its discounted future returns. During training, the agent is encouraged to explore the environment in an attempt to maximize its returns. As training continues and nears its end, exploration is forgone for exploitation of reward maximizing actions via an optimized *policy*. One of the major assumptions in RL is that the information contained in the current state is sufficient to drive a decision – Markov property.

Most reinforcement learning tasks are Markov decision processes (MDPs) and usually have a finite state and action spaces – called finite MDPs. A finite MDP is defined by its state and action sets and by the one-step dynamics of the environment. Given any state s and action a, the probability of each possible pair of next state s' and reward r, is denoted (Sutton and Barto, 2017, see Sec. 3.6):

$$p(s', r|s, a) \doteq Pr\{S_{t+1} = s',\ R_{t+1} = r|\ S_t = s,\ A_t = a\}. \tag{1}$$

The reader is encouraged to consult Sutton and Barto's "Reinforcement Learning: An Introduction" textbook (Sutton and Barto, 2017) for a thorough introduction to the field. In this paper, the probability of getting to the next state s' is based on deterministic equations – it is a deterministic state transitional

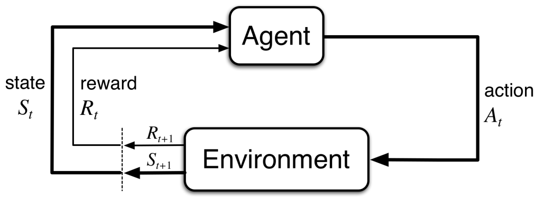

Figure 1: Reinforcement learning process (Sutton and Barto, 2017)

model.

Compared to MPC, RL methods map a state to an action learning a model of the environment internally whereas MPC takes in a model with the current state plus certain predictions and uses optimization to determine the best actions. Once trained, an RL agent will perform faster than an MPC scheme and with a smaller software footprint; however, training the agent to be robust is difficult and remains an open research question.

The second objective of this paper is to conduct a preliminary study to see if RL – on-policy, model-free – can be applied to buildings. Since we do not have a test building to experiment on, we have implemented a general thermal network model framework to emulate a building environment specifically for controls. Future work will consider applying RL to higher fidelity building simulation engines like EnergyPlus – Spawn-of-EnergyPlus (Spawn) with the Functional Markup Interface (FMI) seems to be a promising path, a beta version is expected in 2020 to 2021 (USDOE, 2019) –, TRNSYS or OpenModellica, and lastly on an actual building or experimental setup. In all cases, the simulation models' or experiment's inputs and outputs should be mapped to match the formatting of OpenAI Gym to ease integration.

The contribution of this work is to demonstrate the development and use of a building emulator RL environment. This work is made publicly available[2] where users can develop, implement and then upload their own features.

The advantage of the emulator herein versus higher fidelity models is, firstly, speed. Second, having a test bed where the controller does not see the underlying building model offers a more realistic testing environment since in most controller cases, especially in smaller buildings, there is scarce information about the characteristics of the space or its service systems besides the actuation levels. Having a common test bed, various algorithms and control methodologies can be compared, as well as model-learning algorithms (Kalman filters, Gaussian regression models, optimization-based methods) where the learned model will be used, for example, in MPC.

[1]`https://deepmind.com/blog/alphastar-mastering-real-time-strategy-game-starcraft-ii/`

[2]`https://www.github.com/vderm/SimplifiedBldgControlEnv`

Review

The building controls must be robust to sensor measurement noise and to errors in weather and occupancy predictions to plan the best route or control sequence to apply.

There is a large library of papers modelling buildings for various types of control algorithms and very many on general reinforcement learning problems (20,000 new publications in 2018). However, we were able to identify only a handful that applies reinforcement learning to space conditioning. In their paper, Yamaguchi et al. (2015) have employed Q-Learning RL to an office space with multiple users. The thermal sensations of users were mapped as fuzzy ranges of *hot*, *cold*, or other. The ranges represent our ability to adapt to small temperature changes or to adjusting our clothing levels. The RL policy would try to minimize the overall dissatisfaction of the users. The action space consists of 3 power levels of heating and 3 for cooling. Different methods of comfort aggregation were used for rooms with many occupants. Lastly, they added a penalty for excessive use of cooling but the paper fails to report if it resulted in energy savings or improved comfort. The thermal model seems to be based on Fourier's Law of Heat Conduction.

In Nagy et al. (2018), the authors have not simplified comfort to be *hot* or *cold* but cause a penalty as a function of how far the space temperature is relative to the comfort bounds. The authors claim the equation was based on thermal user comfort literature. The action space in their paper is a discrete set of values between full-off and full-on the agent can select. Overall, the agent is able to maintain a comfortable environment and is able to save 5-10% energy compared to a rule-based approach. They note further improvements can be made.

Recently, Moriyama et al. (2018) have applied RL – trust-region policy optimization (TRPO) learning algorithm – on a data centre to minimize energy use while satisfying temperature constraints. The data centre is modelled in EnergyPlus and EnergyPlus's Energy Management System (EMS) is used to supply external commands while the simulation is running. The inputs (continuous-domain temperatures) and outputs (continuous-domain temperature setpoints and airflow rates) are mapped to match that of OpenAI Gym. The RL reward function is designed to keep the centre's temperature at a fixed setpoint by having small penalties around this value and linearly larger outside a threshold. This would attract the RL agent towards the setpoint by providing a smooth penalty. The authors picked EMS over the FMI since their application was simple enough and EMS was sufficient. The authors have published their source code online and by changing a few files, their framework can be used for other EnergyPlus models. Compared to a simpler equation-based method, EnergyPlus will be slower to train.

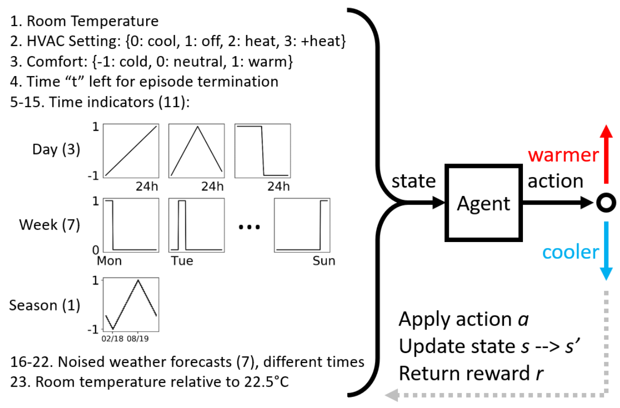
1. Room Temperature
2. HVAC Setting: {0: cool, 1: off, 2: heat, 3: +heat}
3. Comfort: {-1: cold, 0: neutral, 1: warm}
4. Time "t" left for episode termination
5-15. Time indicators (11):
Day (3)
Week (7)
Season (1)
24h
Mon
Tue
Sun
02/18 08/19
16-22. Noised weather forecasts (7), different times
23. Room temperature relative to 22.5°C
state
Agent
warmer
action
cooler
Apply action a
Update state s --> s'
Return reward r

Figure 2: States and actions

Method

For our *environment*, we use a general thermal network based approach. The user can describe their thermal network following the methodology shown in the "Finite Difference Heat Transfer Model" Appendix or simply use one of two predefined network models. The first is a 1st-order model which could represent simple spaces like a condo, a small house or a single office (see Appendix). The second is a 2nd-order model where the heating/cooling is applied to a thermal node that can be different than the occupied space node, such as the case of a radiant slab system. For control purposes, research has shown that simplified models are often suitable to represent the behaviour of a system (Candanedo and Athienitis, 2010; Athienitis and O'Brien, 2015). Our environment implementation is general and allows users to define a building in as much detail as required. A small house can be modelled with a simple model while a sizable office building with varying uses requires more detail.

The environment can run a curriculum of cases to ease learning going from synthetic simple weather and internal loads to more realistic weather read from EnergyPlus "*.epw" weather files. Future work is required to add in realistic and dynamic occupancy behaviour.

The *states* can simply be what a smart thermostat observes plus noisy weather predictions where the further the prediction, the more (Gaussian) noise is added. The uncertainty of weather predictions is based on collected prediction data and compared to what actually happened. These predictions would represent what can readily be obtained from online weather sources. See Figure 2 and last section of the Appendix for simulation parameters, state descriptions and RL agent settings.

The *action* space is discrete: go cooler, stay, go warmer. This way, the agent can learn to either increase or decrease the amount of heating/cooling instead of picking a discrete action over all HVAC settings and translates more to how we tend to explain our comfort, *e.g.* , if a person feels cold, he/she will want the heating increased, if they are still cold, heating should be increased again.

For the *rewards* and penalties (negative rewards), the higher HVAC settings impose a stronger penalty than the low settings. Having the HVAC off is free. The HVAC level penalties do not change during the day, *i.e.* , time-of-use energy pricing is not included, however it could be. To limit excessive cycling of equipment, especially to limit going from heating to cooling within a few timesteps, a rate-of-change penalty is added. When it comes to thermal comfort, the agent needs to maintain the room temperature within the setpoint bands otherwise a linear penalty is applied. The comfort penalty function of Nagy et al. (2018) was also implemented but not used as the linear penalty yielded better results. No terminal reward is given at the end of the episode. Details in Table 5 in the Appendix.

The *agent* selected in this paper uses the proximal policy optimization (PPO) method of learning. PPO was developed by OpenAI (Schulman et al., 2017) and in a blog post[3], they say "[PPO performs] comparably or better than state-of-the-art approaches while being much simpler to implement and tune" and is now one of the go-to algorithms in RL. Table 3 in the Appendix describes all parameter settings. PPO is a policy gradient method and it maps states to actions directly instead of going through a value or action-value function; we suggest the reader to consult the chapter about policy gradient methods in Sutton and Barto's book (chapter 13 as of the date of this publication).

The environment is customizable. The reward or penalty schedule can be changed to correspond to time-of-use electrical pricing, or a demand response signal, or to improve on-site consumption of renewable energy generation. The HVAC system can have as many operating modes as desired, but they need to be input in ascending order of power. Temperature limits can be adjusted and adapted based on occupant feedback or adaptive models. The agent can also be swapped for another.

Results

To compare the performance of the reinforcement learning agent, we propose a bang-bang and proportional-integral (PI) controller that correspond to what is currently present in most homes. The bang-bang controller will have two stages for heating and one stage for cooling. In a heating application, if the temperature drops below a set temperature, stage 1 will be activated until the room temperature passes a threshold and then it will turn off to reduce cycling of the equipment. If the temperature of the room instead keeps dropping, a second threshold is used to

[3]`https://blog.openai.com/openai-baselines-ppo`

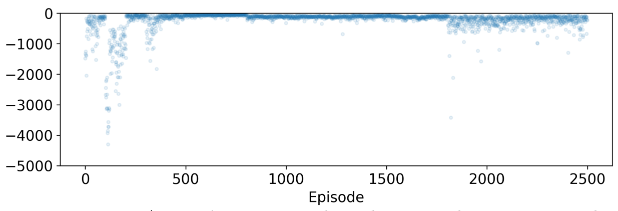

Figure 3: Agent's accumulated rewards per episode

activate a second stage of heating. Cooling is applied in a similar manner but for when the space is hot.

The PI controller will supply heat proportional to the setpoint error $e(t)$ – the difference between the current and setpoint temperatures. The integral term is used to correct for oscillatory behaviour. In our case here where the HVAC system works in discrete stages, the HVAC stage that comes closest to the output from the PI controller will be used with an additional deadband to, again, reduce cycling of the equipment. The PI output is given by:

$$\text{output}(t) = K_p\, e(t) + K_i \int_0^t e(t')dt' \qquad (2)$$

Where K_p and K_i are the proportional and integral factors that are selected to improve response and limit oscillatory behaviour. These factors were chosen based on heuristic rules and iteratively. Bang-bang and PI controller parameters are given in Table 4 in the Appendix.

The reinforcement learning agent is trained on the environment for 2500 episodes, 200 on synthetic cases, 1600 of which on random days of the year and the remainder on random weeks of the year. To test learning, the weather file can be changed to assure memorization is not occurring. Also, the neural network *brain* of the agent is not deep enough to truly memorize the weather file. In this paper, a typical house in Montreal is considered without and with a radiant floor system; Montreal weather data is used.

The agent's learning can be tracked by looking at its accumulated rewards per episode as training progresses, see Figure 3. Here, we observe an increase in the collected rewards until it seems to plateau and training is stopped to reduce over-fitting. After 1800 episodes, the learning curriculum is switched and we observe a strong but brief decline in the rewards. As a cautionary note, over-training an agent on simple and synthetic cases can lead to a collapse in the weights of the neural network *brain* and learning irreversibly ceases; the agent needs to be then reinitialized.

Heating Scenario

Beginning with the heating cases. We have selected a very cold week in winter (see bottom-left subplot of Figure 5 for the ambient temperature), to compare how conventional controls compare to the RL agent. As can be seen from Figure 4, left, the bang-bang and

PI controllers try to maintain the lower limit temperature (setpoint was 20.5-21°C), the applied actions are representative of their design. The PI temperature over time is slowly moving upwards due to the integral portion of the controller. The RL controller, due to the set penalties, tries to maintain the temperature within the comfort boundaries and minimizes toggling of the HVAC system. The penalty weights can be tuned to instead reduce energy consumed and the room temperature would hover nearer to the lower limit. A grid-search or Bayesian optimization can be performed to determine parameter values to fine-tune the RL controller performance to users' needs.

Cooling Scenario

In the cooling case, a hot summer week is selected (see bottom-right subplot of Figure 5). Compared to the conventional controllers, the RL case performs worst (see Figure 4, right). During the first day, the RL agent keeps the cooling off to minimize its HVAC toggle penalty which led to warmer space temperatures. The following days, it recovers and maintains comfort, but tends to hug the lower comfort boundary which requires more cooling energy.

Radiant Floor System

In the previous two cases, the 1st-order model was used where the HVAC system supplies the space directly. To study if the RL agent makes use of the future weather predictions returned by the environment, a case where the heating or cooling is supplied indirectly such as heating or cooling a hydronic radiant floor slab should be analysed. We have trained the RL agent on the 2nd-order model to see if it will utilize pre-heating and pre-cooling strategies. Results shown in Figure 5 for both winter and summer cases. Model parameters are listed in Table 2 in the Appendix.

We do see the temperature being maintained between the setpoints. Looking closely at the HVAC states and the ambient temperature – the bottom two subplots – we see pre-cooling for the summer case at around the 84th and 104th hours. The cooling is turned on before the peak temperature. The heating is turned on briefly at the 144th hour. A stronger penalty can be incorporated to ensure the HVAC system does not go from cooling to heating within a few hours; there is only a toggling penalty at this moment to avoid the use of both heating and cooling during the same day. In the winter heating case, the exterior ambient temperature is rather flat throughout the day and there is no advantage in pre-heating; heating should be supplied continuously.

Discussion and Future Work

In this study, we fixed the emulator model parameters to be more or less a typical house in Montreal based on our prior experience. In an application, the environment parameters that characterize the building itself can in fact be estimated and updated overtime using a Kalman filter approach based on the response of the building (Radecki and Hencey, 2012). The RL agent would then rely on the updated environment to train and would send commands to the actual building. This approach would effectively be closer to a model-based method and offers a low-disruptive control application. The reward function penalizes HVAC toggles, however, for a house in winter, having both heating and cooling on the same day should be disallowed by strongly penalizing this behaviour. For an office space, heating and cooling in the same day is possible.

As mentioned earlier, Ecobee has a "donate your data" program (Ecobee, 2018). This anonymized dataset can be used to determine environment parameters – one set per house. We can use these parameters and create a building environment instance per house. We can then train an agent on many different houses to come up with a "good enough" policy that can be mass-deployed on every intelligent thermostat. This policy would then be specialized to its house over time and discomfort reward signals would come directly from the occupants. Each person having a different thermal preference, the agent would eventually adapt to their comfort requirements.

Additionally, having a large set of building environment instances, we can essentially simulate the behaviour of a neighbourhood and perform community energy strategies/studies, *e.g.* Bucking and Dermardiros (2018). These simulated communities can help utilities test out various incentives and load management strategies.

One of the limitations of RL is that it tends to require an enormous amount of episodes and data to learn an optimal strategy/policy. This is because the RL agent not only needs to understand the environment it is in – effectively creating an approximate representation of the environment – but to also apply optimal actions. Action selection in itself is a difficult task. Recent work applies MPC in RL to improve action selection and/or to create an approximate physics-based model of the environment, *e.g.* Kamthe and Deisenroth (2017). Having introduced more structure to the problem, the agent can learn the optimal policy with greater ease which translates to a more effective use of episodic experience and in robust actions.

Conclusion

The main contribution of this work is the development of an open-source Python emulator with structure matching that of the OpenAI Gym environments. By relying on a standardized environment, cutting-edge reinforcement learning based methods can be directly applied to building service system controls. From the point of view of the controller,

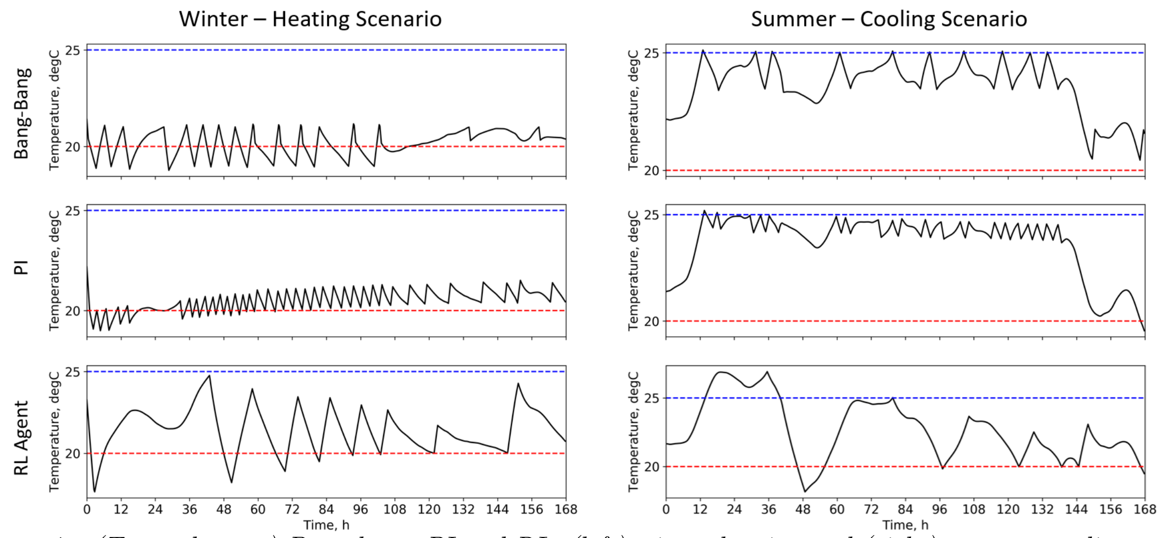

Figure 4: (Top to bottom) Bang-bang, PI and RL: (left) winter heating and (right) summer cooling cases

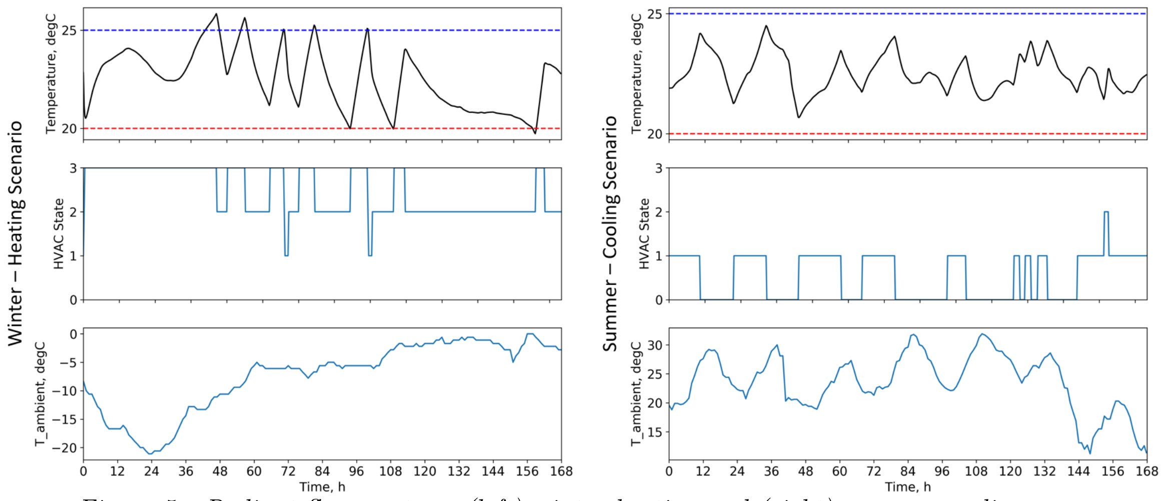

Figure 5: Radiant floor system: (left) winter heating and (right) summer cooling cases

the emulator corresponds to a model-less test bench where various control strategies and model-learning algorithms can be tested and compared. The emulator is based on a thermal network model representation. Two predefined models are included and a generic model where the user can design their system with as much detail as required. The emulator uses standard EnergyPlus "*.epw" weather files which can be swapped. Dynamic and realistic occupancy behaviour remains as future work. Another future work would be to partially train the agent on our environment and then move it to a more realistic/detailed model following the EnergyPlus EMS OpenAI Gym wrapper methodology proposed by Moriyama et al. (2018).

As a second contribution and case study, a PPO RL agent was trained on the environment on a curriculum of cases: random days in a year and random weeks in a year. The trained agent was compared to conventional bang-bang and proportional-integral controls. The RL agent tries to apply control actions to maximize its rewards. Here, comfort and toggling the HVAC equipment had a relatively large attributed penalty compared to energy use and so the resulting agent would minimize toggling the equipment and having the space temperature fluctuate within the comfort bounds. Given this learned behaviour, equipment life is likely to be extended compared to bang-bang and PI controllers.

The emulator environment and examples are uploaded to our GitHub repository[4].

References

Athienitis, A. K. E. and W. E. O'Brien (2015). *Modeling, Design, and Optimization of Net-Zero Energy Buildings* (1 ed.). Berlin, Germany: Ernst & Sohn.

Bucking, S. and V. Dermardiros (2018). Distributed evolutionary algorithm for co-optimization of building and district systems for early community energy masterplanning. *Applied Soft Computing 63*, 14–22.

[4]`https://www.github.com/vderm/SimplifiedBldgControlEnv`

Candanedo, J. A. and A. K. Athienitis (2010). Simplified linear models for predictive control of advanced solar homes with passive and active thermal storage.

Denholm, P. and M. Hand (2011). Grid flexibility and storage required to achieve very high penetration of variable renewable electricity. *Energy Policy 39*(3), 1817–1830.

Dhariwal, P., C. Hesse, O. Klimov, A. Nichol, M. Plappert, A. Radford, J. Schulman, S. Sidor, Y. Wu, and P. Zhokhov (2017). OpenAI Baselines. `https://github.com/openai/baselines`.

Ecobee (2018). Donate your Data. `https://www.ecobee.com/donateyourdata/`.

Jensen, S. O., A. Marszal-Pomianowska, R. Lollini, W. Pasut, A. Knotzer, P. Engelmann, A. Stafford, and G. Reynders (2017). IEA EBC Annex 67 Energy Flexible Buildings. *Energy and Buildings 155*, 25–34.

Kamthe, S. and M. P. Deisenroth (2017). Data-Efficient Reinforcement Learning with Probabilistic Model Predictive Control. *arXiv:1706.06491 [cs, stat]*. arXiv: 1706.06491.

Morales, J. M., A. J. Conejo, H. Madsen, P. Pinson, and M. Zugno (2014). *Integrating Renewables in Electricity Markets: Operational Problems*, Volume 205. Springer Science & Business Media.

Moriyama, T., G. De Magistris, M. Tatsubori, T.-H. Pham, A. Munawar, and R. Tachibana (2018). Reinforcement learning testbed for power-consumption optimization.

Nagy, A., H. Kazmi, F. Cheaib, and J. Driesen (2018). Deep Reinforcement Learning for Optimal Control of Space Heating. pp. 8.

Nhede, N. (2018). Role of smart thermostats to grow in future utility operations. `https://www.metering.com/features-analysis/smart-thermostats-market-analyses/`.

OpenAI (2018). OpenAI Five. `https://blog.openai.com/openai-five/`.

Radecki, P. and B. Hencey (2012). Online building thermal parameter estimation via unscented Kalman filtering. In *American Control Conference (ACC), 2012*, pp. 3056–3062. IEEE.

Schaarschmidt, M., A. Kuhnle, and K. Fricke (2017). TensorForce: A TensorFlow library for applied reinforcement learning. `https://github.com/reinforceio/tensorforce`.

Schulman, J., F. Wolski, P. Dhariwal, A. Radford, and O. Klimov (2017). Proximal Policy Optimization Algorithms. *arXiv:1707.06347 [cs]*. arXiv: 1707.06347.

Shaikh, P. H., N. B. M. Nor, P. Nallagownden, I. Elamvazuthi, and T. Ibrahim (2014). A review on optimized control systems for building energy and comfort management of smart sustainable buildings. *Renewable and Sustainable Energy Reviews 34*, 409–429. R.

Silver, D., J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. van den Driessche, T. Graepel, and D. Hassabis (2017). Mastering the game of Go without human knowledge. *Nature 550*(7676), 354–359.

Sutton, R. and A. G. Barto (2017). *Reinforcement Learning: An Introduction* (2 ed.). Cambridge, MA: MIT Press.

Thomas, N. (2017). DeepMind and National Grid in AI talks to balance energy supply. `https://www.ft.com/content/27c8aea0-06a9-11e7-97d1-5e720a26771b`.

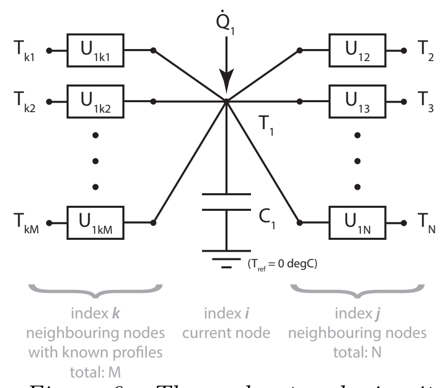

Figure 6: Thermal network circuit

USDOE (2019). Spawn-of-EnergyPlus. `https://www.energy.gov/eere/buildings/downloads/spawn-energyplus-spawn`.

Yamaguchi, Y., N. Shigei, and H. Miyajima (2015). Air Conditioning Control System Learning Sensory Scale Based on Reinforcement Learning. In *Proceedings of the International MultiConference of Engineers and Computer Scientists*, Volume 1.

Acknowledgements

We would like to acknowledge the input from Doina Precup, Pierre-Luc Bacon and Maziar Gomrokchi from McGill's Reasoning and Learning Lab. We would also acknowledge Tawfeeq Jawwar for multiple discussions. And finally, we are thankful for all the high quality open source resources made available online, be it Sutton and Barto's RL textbook, RL frameworks such as OpenAI Gym and Baselines, Tensorforce, thorough blog posts and the rich library of Python libraries.

Vasken Dermardiros would like to acknowledge the financial support received from the NSERC Alexander Graham Bell Doctorate Scholarship, the Hydro-Qubec Engineering and Computer Science (ENCS) Entrance Scholarship, the Professor Hugh McQueen Award of Excellence, and the Concordia Faculty of ENCS Graduate Scholarship.

Appendix

Finite Difference Heat Transfer Model

For this project, the implicit finite difference approximation (first-order backward in time Taylor approximation) to Fourier's heat transfer equations was used. Generalizing for a system, the equation can be written in a matrix form as shown in Equation 3 where, nN is the number of nodes, and, nM is the number of nodes with known temperatures (boundary condition).

As an example where $i = 1$, we can draw the thermal network in Figure 6.

Where, U_{ij} is the conductance between nodes i and j equal to $\frac{kA}{dx}$ for conductance, $h_{conv}A$ for convection and $h_{rad}A$ for radiation, $\frac{W}{K}$; C is the capacitance of

$$\begin{bmatrix} \sum_j^{nN} U_{1j} + \sum_k^{nM} U_{1k} + \frac{C_1}{\Delta t} & -U_{12} & \dots & -U_{1nN} \\ \vdots & \vdots & \ddots & \vdots \\ -U_{nN1} & -U_{nN2} & \dots & \sum_j^{nN} U_{nNj} + \sum_k^{nM} U_{nNk} + \frac{C_{nN}}{\Delta t} \end{bmatrix} \begin{Bmatrix} T_1 \\ \vdots \\ T_{nN} \end{Bmatrix}^{t+1} = \begin{Bmatrix} \dot{Q}_1 + \sum_k^{nM} (U_{1kk} T_{kk}) + \frac{C_1}{\Delta t} T_1^t \\ \vdots \\ \dot{Q}_{nN} + \sum_k^{nM} (U_{nNkk} T_{kk}) + \frac{C_{nN}}{\Delta t} T_{nN}^t \end{Bmatrix}, \quad (3)$$

node i equal to $\rho c_p A dx$, $\frac{J}{K}$; $\dot{Q}$ is the heat flow into the node, W; and, Δt is the timestep, s.

1st and 2nd-Order Pre-Defined Models

In the experiments reported in this paper, we have relied on a simple 1st-order pre-defined model called "F1C1", see Figure 7. Its effective capacitance and conductance values are based on typical values for a reasonably well-insulated single family home in Montreal – based on experience, see Table 1.

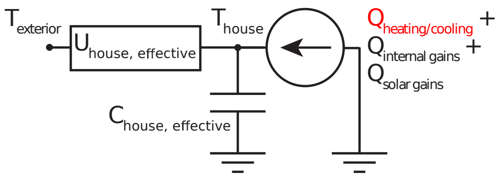

Figure 7: Thermal network circuit for F1C1 model. Heat input in red is controlled by the reinforcement learning agent.

Table 1: 1st-order building model "F1C1" parameters

Parameter	Value
C, effective, J/K	$12 \cdot 10^6$
U, effective, W/K	250
Δt, s	900

We have also successfully tested the environment using higher order models such as the case of having the heating/cooling system applying energy on a different node than that of where the thermostat is measuring, *e.g.* , a space heated and cooled using a radiant slab system, or via a thermal storage source. Table 2 summarizes the 2nd-order model parameters.

Table 2: 2nd-order building model "F1U1C2" parameters

Parameter	Value
C, room, J/K	$2 \cdot 10^6$
C, slab, J/K	$10 \cdot 10^6$
U, to exterior, W/K	250
U, slab to room, W/K	2775
Δt, s	900

Simulation and RL Agent Settings

Table 3 summarizes the reinforcement learning agent parameter settings. Table 4 describes the bang-bang and PI controller settings. Table 5 summarizes the simulation parameter.

Table 3: Reinforcement learning agent parameters

Parameter	Value or Description
Agent type	Proximal policy optimization (PPO)
Library	Tensorforce v.0.4.3
Agent network	Dense 64 units + LSTM 64 units
Optimizer	ADAM, learning rate: 1e-3
Other settings	As per Quickstart tutorial, link: `https://bit.ly/2Tgi21i`

Table 4: Bang-bang and PI controller parameters; Heating Stage 1 and 2 correspond to HVAC Setting 2 and 3. Cooling is HVAC Setting 0

Scenario	Controller	Parameter	Value
Heating	Bang-bang	ON, Stage 1	20.5
		OFF, Stage 1	21.5
		ON, Stage 2	19
		OFF, Stage 2	21
	PI	Setpoint	21
		Kp	5000
		Ki	20
		Deadband	2000W
Cooling	Bang-bang	ON	25
		OFF	23.5
	PI	Setpoint	24
		Kp	3000
		Ki	10
		Deadband	1000W

Table 5: Simulation parameters

Env. Perturbations	Description
Curriculum, episodes	200 synthetic + 1600 daily random + 700 weekly random
Temperature, boundary	Montreal *.epw" file
Int. gains, episode	Uniform(100, 800)
Solar gains	"*.epw" file * 0.5 (occlusion)
Env. Settings	**Description**
T° initial, °C	N(22.5,0.5...2.0)
T° comfort limits, °C	Low: 20, high: 25
HVAC setting 0	-3000W (cooling), cost: -2
HVAC setting 1	0W (off), cost: 0
HVAC setting 2	5000W (heating), cost: -2
HVAC setting 3	10,000W (heating), cost: -4
HVAC initial setting	Setting 2, off
HVAC cost multiplier	0.2
Action toggle penalty	-0.8
Reward, termination	0