Poster Abstract: COBS: COmprehensive Building Simulator

Tianyu Zhang
University of Alberta
Edmonton, AB, Canada
tzhang6@ualberta.ca

Omid Ardakanian
University of Alberta
Edmonton, AB, Canada
ardakanian@ualberta.ca

ABSTRACT

This poster abstract describes the design and implementation of an open-source building co-simulation platform which integrates various models, simulators, and controllers (e.g., HVAC, lighting, and window blinds), enabling comprehensive evaluation of occupant-centric building controls. Using proper locking, it interacts with EnergyPlus in each time step to access and modify the simulation model object before running simulation in the next time step. The interface is designed similar to OpenAI Gym so that environments that are already written for Gym can be used with this platform. The proposed platform utilizes a queueing network simulator to generate realistic movement traces for occupants and simulate actions in a probabilistic fashion.

CCS CONCEPTS

• **Software and its engineering** → **Designing software**; • **Computer systems organization** → *Embedded and cyber-physical systems.*

ACM Reference Format:
Tianyu Zhang and Omid Ardakanian. 2020. Poster Abstract: COBS: COmprehensive Building Simulator. In *The 7th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation (BuildSys '20), November 18–20, 2020, Virtual Event, Japan.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3408308.3431119

1 INTRODUCTION

Buildings account for 39% of the total energy consumption in the U.S. [1]. A large fraction of this energy is consumed to maintain thermal and visual comfort of the occupants. This together with the fact that people in North America spend 87% of their time indoors [6] emphasizes the importance of designing energy-efficient building controls that incorporate information about occupants' presence and actions to maximize their comfort. However, with the large number of control knobs that exist in a typical commercial building and stringent comfort requirements, evaluating different control policies on top of buildings with real occupants is onerous and costly. Consequently, most control policies today are evaluated for a short period of time on a few test buildings.

Executing control policies in a simulated environment can help overcome the barriers to real-world deployment of a new control policy by enabling comprehensive evaluation of this policy in buildings with various sizes and occupancy schedules, located in different climates. Furthermore, policy evaluation in a simulated environment is useful for the design of reinforcement-learning (RL) controllers that improve a policy in an iterative fashion and benefit from offline training. Existing building energy simulators, such as EnergyPlus, can provide an accurate building energy analysis over multiple days within seconds, but they suffer from two fundamental problems. First, they require the full control policy before running a simulation. This prevents users from writing code that interfaces with other simulators and incorporating external models in their control algorithm. Second, they only focus on the number of people present in each zone, neglecting effects of their actions on the environment. To simulate occupants' actions, it is necessary to track each occupant inside the building and simulate actions (e.g., turning lights on, opening blinds, adjusting temperature setpoints) conditioned on their location. This is crucial as in most cases occupants must be in the proximity of an actuator or a control panel in order to operate it. Due to these shortcomings, recent studies [4] build standalone simulators using BCVTB [7] to generate data for offline learning.

In this paper, we introduce COBS[1], an open-source and modular co-simulation platform in Python which is designed to support fine-grained control over the states of multiple building subsystems (which can be modelled separately) in each step of simulation. It provides the ability to include and exchange data between multiple models, allows benchmarking control algorithms across many buildings, and facilitates online learning. Additionally, COBS utilizes an occupancy simulator which generates realistic individual trajectories and samples interactions between occupants and building subsystems from conditional probability distributions. These probabilities can be learned from datasets that capture how occupants interact with building interfaces. We briefly describe COBS's architecture in the next section.

2 ARCHITECTURE

COBS is developed with three design goals. First, it needs to return the observed building state at each time step, and execute user-specified actions to update this state for the next time slot. This is imperative for implementing learning-based control algorithms. Second, it must provide a simple interface for the inclusion of state estimators and predictive models (for room temperature, occupancy, solar radiation). Adding predictions to the state is essential for proactive control of building subsystems. Third, it needs to interface with data synthesizers to obtain traces, e.g., for occupant movements and actions. This is in contrast to existing simulation packages, such as EnergyPlus, which take as input pre-defined schedules for occupants, windows, lights, etc.

[1]COBS can be downloaded from https://github.com/sustainable-computing/COBS/. The documentation is available at https://cobs-platform.github.io.

BuildSys '20, November 18–20, 2020, Virtual Event, Japan

ACM ISBN 978-1-4503-8061-4/20/11.
https://doi.org/10.1145/3408308.3431119

Figure 1 shows the overall architecture of COBS. It takes the building IDF file created by modelling software, such as SketchUp, OpenStudio, and EnergyPlus, and combines it with the output of an occupancy simulator which determines the location of each occupant and any actions they may perform in that location. The resulting model is used to implement actions in the event queue and simulate the building state using EnergyPlus. The platform takes advantage of a priority scheduling algorithm to schedule various time-stamped events. The simulated building state is then modified and augmented by several estimation and prediction models. This updated state is then used for reward calculation (given a user-defined function) and sent to the control agent. All actions, rewards, and modified state variables for each time slot are stored in a replay buffer for ease of access in the future.

Our framework enables the agent to learn an optimal control policy through direct interaction with the simulated building environment. This is particularly useful for implementing RL algorithms to optimally control HVAC and lighting systems or window blinds, an area that has received increasing attention in recent years [5]. Unlike the environment used in RL, our model provides the agent with not only historical and real-time data but also future predictions. This improvement makes the design of RL algorithms easier and opens the door to the integration of several models with the building control agent. The platform consists of three main components described below.

- **Model Class** consists of a replay buffer, a building model in IDF format, and several models for estimating, predicting, and modifying the state returned by EnergyPlus. The platform provides methods like `reset` and `step` following the same structure as the OpenAI Gym [3]. Thus, RL agents written using Gym can be ported to our platform with small changes. The platform uses multiprocessing and locking mechanisms to support real-time interaction between agents and EnergyPlus, thereby ensuring the action is implemented before simulating the next state.
- **EventQueue Class** uses a priority queue to store and schedule all actions at specified times. The queue determines the order of execution of different actions in each time slot according to their priority and insertion time. Agents and predictive models can access the queue to retrieve future events and make decisions based on them if needed.
- **OccupancyGenerator Class** exploits a queueing network simulator to produce realistic occupancy schedules. The queueing network is constructed according to the floorplan of the building, probabiilties of visiting different spaces upon leaving a space, and the average time spent in each space (terminal zone). We elaborate on this process in Section 3.

3 OCCUPANTS' MOVEMENTS AND ACTIONS

Several studies suggest that occupants behaviour can greatly affect the energy use and number of comfort violations [2]. For instance, occupants may open/close windows and doors, resulting in a considerable change in the carbon-dioxide concentration, air flow, and room temperature. Control systems respond to this change in different ways to maintain the temperature around the setpoint and meet comfort requirements. This implies that using a fixed occupancy schedule and neglecting actions to evaluate control policies may lead to different conclusions.

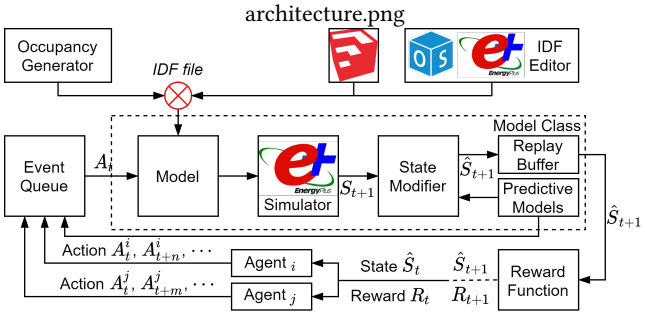

Figure 1: Overall Architecture

To address the challenge of collecting occupancy data, including occupant presence and actions, we propose an occupancy generator that draws on queueing theory to generate trajectories for occupants. Furthermore, we use the synthesized occupant trajectory to simulate the occupant actions based on control knobs that exist in the room where the occupant is and the conditional probability provided to the generator in the form of a JSON file.

We treat each occupant as a job in the queueing network and each zone as a FCFS queueing system with infinite servers and exponentially distributed service times. Therefore, each zone is an $M/M/\infty$ queue, and the whole building can be modelled as an open queueing network comprised of N queues which are connected based on the floorplan of the building. Occupants arrive to the building following a Poisson process with a rate that depends on time of the day. Concretely, the arrival rate is relatively higher in the morning when people are expected to go to work.

The stay time in each zone depends on its function for each occupant. Occupants stay longer in their designated office space and shorter in other spaces in the building. Movements inside the building are governed by a probability which is higher for returning to their office and lower for visiting other spaces upon leaving a space. The time spent moving between spaces is also considered. We assume the service in this queueing network can be interrupted by a number of events, including the lunchtime and start of a meeting.

After simulating the locations of each occupant for a given simulation period, we calculate the total number of occupants in each zone and store it in the model. The occupancy location is then used to simulate occupants' actions; they must be simulated for each time slot because their occurrence may depend on the current state of the building. Hence, in each time step, we filter out infeasible actions in all occupied zones and decide whether to simulate an action by sampling from a conditional probability distribution.

REFERENCES

[1] U.S. Energy Information Administration. 2019. *Annual Energy Review 2019.* U.S. Government Printing Office, Chapter Energy consumption estimates by sector.
[2] Rune Andersen et al. 2009. Survey of occupant behaviour and control of indoor environment in Danish dwellings. *Energy and Buildings* 41, 1 (2009), 11–16.
[3] Greg Brockman et al. 2016. OpenAI Gym. arXiv:1606.01540
[4] Bingqing Chen et al. 2019. Gnu-rl: A precocial reinforcement learning solution for building HVAC control using a differentiable MPC policy. In *Proc. ACM BuildSys.* ACM, 316–325.
[5] Konstantinos Dalamagkidis et al. 2007. Reinforcement learning for energy conservation and comfort in buildings. *Building and environment* 42, 7 (2007), 2686–2698.
[6] Neil Klepeis et al. 2001. The National Human Activity Pattern Survey (NHAPS). *Journal of Exposure Science & Environmental Epidemiology* 11, 3 (2001), 231–252.
[7] Michael Wetter. 2011. Co-simulation of building energy and control systems with the Building Controls Virtual Test Bed. *Journal of Building Performance Simulation* 4, 3 (2011), 185–203.