Large Neighborhood Search for Temperature Control with Demand Response

Edward Lam[1,2][0000−0002−4485−5014], Frits de Nijs[1][0000−0003−4466−2447], Peter J. Stuckey[1][0000−0003−2186−0459], Donald Azuatalam[1][0000−0002−2149−8122], and Ariel Liebman[1][0000−0002−5679−4140]

[1] Monash University, Melbourne, Australia
[2] CSIRO Data61, Melbourne, Australia
{edward.lam,frits.nijs,peter.stuckey,
donald.azuatalam,ariel.liebman}@monash.edu

Abstract. Demand response is a control problem that optimizes the operation of electrical loads subject to limits on power consumption during times of low power supply or extreme power demand. This paper studies the demand response problem for centrally controlling the space conditioning systems of several buildings connected to a microgrid. The paper develops a mixed integer quadratic programming model that encodes trained deep neural networks that approximate the temperature transition functions. The model is solved using standard branch-and-bound and a large neighborhood search within a mathematical programming solver and a constraint programming solver. Empirical results demonstrate that the large neighborhood search coupled to a constraint programming solver scales substantially better than the other methods.

Keywords: Sustainability · Energy systems · Power systems · Control · Smart grid · Microgrid · Large neighborhood search · Local search

1 Introduction

Electricity utilities are required to ensure supply-demand balance throughout the power grid since any mismatch can cause voltage or system frequency instability, resulting in loss of supply or system blackouts. Supply-demand balance will become increasingly difficult with increased investment in uncontrollable renewable generations such as solar and wind. To more effectively and cost-efficiently safeguard system reliability, utilities are increasingly deploying demand response to cater for unforeseen or difficult circumstances in their network, such as extreme weather, by motivating customers to decrease demand at certain times or to shift their consumption to off-peak periods.

Figure 1 shows four example power profiles of a building over a typical day. Normal power consumption, peaking at nearly 8 kW, is shown in black. As appliances and the building's insulation receive upgrades in energy efficiency, the load profile is expected to shift from the black line to the green line, which peaks

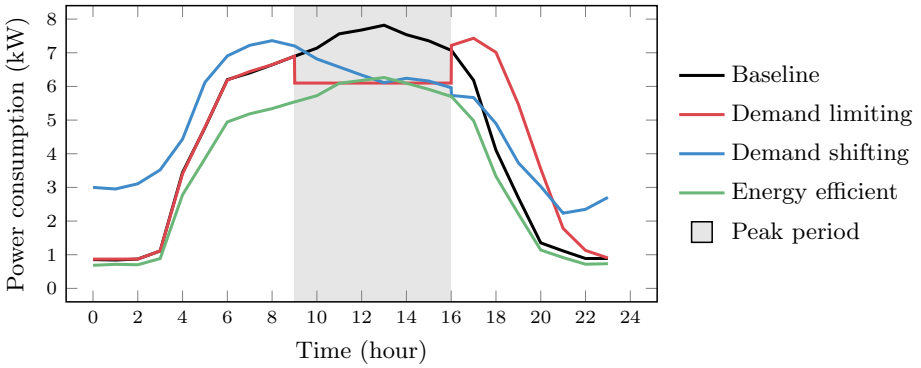

Fig. 1. Load shapes of a building, with and without demand response [13].

at just over 6 kW. In the meantime, utilities promote grid reliability and stability by allowing building management to either curtail power use (shown in red) or shift power usage to earlier parts of the day (shown in blue) when the temperature is cooler; preserving power for heating, ventilation and air conditioning (HVAC) systems in the warmer parts of the day.

Utilities target customers with large controllable devices with flexible usage for participation in demand response. Examples of such devices include thermostatically-controlled loads (e.g., HVAC systems and electric water heaters) and shiftable appliances (e.g., dishwashers, washing machines and dryers). Since space conditioning accounts for a large proportion of building energy use, especially in commercial buildings, HVAC loads are excellent candidates for demand response. Moreover, slight temperature changes do not immediately impact occupant comfort because of the thermal storage effect of buildings [13], a feature that can be further enhanced through building upgrades such as hot water storage tanks and phase-change materials [3].

Even though a manual approach to demand response can be tediously implemented, HVAC systems in modern commercial buildings can be controlled by automation systems. This paper develops strategies to jointly control the HVAC systems of several buildings connected to a microgrid such that their temperatures are maintained within acceptable comfort bounds and all power restrictions are adhered to. We make use of a deep neural network to automatically learn a transition function for each of the buildings under control, as a proxy for the behavior governed by physical laws. This paper shows how these neural network transition functions can be encoded in a mixed integer quadratic programming (MIQP) model that decides on the operating mode of the HVAC system in all connected buildings within a given planning horizon.

The model is implemented in the constraint programming (CP) solver Gecode and the mathematical programming (MP) solver Gurobi. A large neighborhood search (LNS) is also implemented in the two exact solvers for finding local improvements. Empirical results show that the best solutions are found using LNS in Gecode. The remainder of the paper explores these results in detail.

2 Related Work

HVAC systems account for approximately 50% of total building energy consumption [15]. Their large energy footprint, combined with their flexibility due to thermal inertia, has resulted in significant efforts in developing methods to unlock their potential for demand response, including multiple field tests [12]. Unfortunately, these methods are largely market-based, controlling the HVACs through price signals rather than computing a globally optimal joint schedule. Given the true model dynamics for each building, the globally optimal joint schedule can be approximated efficiently by Lagrangian relaxation methods such as column generation [14]. However, the use of relaxations means that constraints are not guaranteed to be satisfied, unless schedules are corrected by re-planning in an on-line fashion. This requires that the agents are in constant communication, which is a potential system vulnerability. The optimization methods proposed in this paper avoid this issue by computing feasible schedules a priori.

When accurate models are not available, one solution is to directly learn a controller by interacting with the system, through reinforcement learning. This approach has seen wide application to demand-response problems [20]. However, the vast majority of these approaches apply only to single-agent problems; learning to control the joint dynamics directly is highly intractable due to the curse of dimensionality on the exponential growth of the state and action space, while multi-agent reinforcement learning does not have a clear pathway to impose global constraints. In addition, reinforcement learning requires the reward function coefficients to be fixed a priori, so changes in a user's comfort preferences necessitate learning a new solution. To avoid these challenges, we propose separating learning the transition function from planning the HVAC schedules.

We adapt a variety of techniques in the combinatorial optimization literature to the demand response problem. The use of artificial neural networks (ANNs) as approximations of complex processes in optimization models has been championed in [4]. They develop a Neuron global constraint and bounds-consistent filtering algorithms, which are evaluated in controlling the temperature of computer chips under various workloads. In [2], an ANN is encoded in a mixed integer linear programming model using facet-defining (i.e., tightest possible) constraints. However, they show that the new constraints are outperformed by a simple big-M encoding in practice.

A learned transition function in the form of a ReLU-ANN can be encoded directly as a mixed-integer linear program [18]. In their work, the authors develop valid inequalities that allow sparsifying the encoding of an ANN, resulting in significant speed-up. Among other domains, they apply their approach to a multi-zone HVAC control problem, with an objective to minimize cost of keeping occupants comfortable. Due to the complexity of the mixed-integer linear programming, only relatively small instances with short horizons can be solved optimally, requiring on-line planning. Compared to their work, we consider demand-response in a larger multi-building setting, which we show to be intractable to solve directly, requiring the use of our local search procedures to solve in reasonable time.

3 The Problem

This section describes the problem and then models it using MIQP.

3.1 Motivation

Monash University has committed to achieving net zero emissions by 2030. It is investing AUD$135 million to increase the energy efficiency of its operations, electrify its buildings, and transition to renewable electricity through on-site solar generation and off-site power purchase agreements. The project aims to develop solutions to the university's operations, and to serve as a testbed for scientific and engineering studies by integrating its education, research and industry activities with its built environment. This approach has been recognized globally, having won the United Nation's Momentum for Change Award in 2018.

One component of the project is to redevelop a portion of the main university campus at Clayton, Victoria, Australia into an energy-efficient microgrid. The project has seen the installation of distributed energy resources including medium-scale solar photovoltaic generation, precinct-scale batteries and electric vehicles that can supply the microgrid while stationary. Buildings with varied usage characteristics, including commercial buildings and student residences, are also refurbished for improved energy efficiency and their HVAC systems, if not entirely deprecated by gains in energy efficiency, are upgraded for autonomous control by the microgrid.

The purpose of this paper is to study merely one small portion of a real-world microgrid project. This paper considers a simplified problem abstracted from the problem of jointly controlling the internal air temperature of several buildings subject to occupant comfort and demand response. For simplicity, every building is assumed to have one independent HVAC system that influences its future indoor temperature by operating in one of three modes: off, cooling or heating. However, the approach naturally generalizes to multiple control zones within a building.

3.2 The State Transition Function

The change in temperature from one timestep to the next (i.e., the state transition function) is governed by physical laws. The future indoor temperatures are a function of the current indoor and outdoor air temperature, the internal mass temperature (a measure from the occupants and furniture), the internal humidity, the building insulation, the solar irradiance as well as many other factors. Ideally, the parameters of the physical laws are estimated from data collected in the environment, and then a physical model is encoded into an optimization model. However, such physical models are typically non-linear, making them hard to encode and solve, and high-accuracy simulation is computationally expensive. Furthermore, estimating the parameters from data already entails the use of machine learning. Therefore, this paper argues for a data-driven approach to directly estimate the entire transition function using a deep neural network,

rather than estimating the building parameters in a finite difference discretization of the governing differential equations.

The transition function is approximated by a multi-layer perceptron (MLP), also called a vanilla feedforward neural network. MLPs are a basic workhorse in machine learning. This section briefly review MLPs. For a formal treatment, readers are recommended to consult the definitive textbook [10].

An MLP $F_{\boldsymbol{\theta}} : \mathbb{R}^n \to \mathbb{R}^m$ with parameters $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_k)$ for some $k \in \mathbb{N}$ is a function that maps a real-valued n-dimensional vector input to a real-valued m-dimensional vector output. Consider a data set $\mathcal{D}$ consisting of pairs of real-valued vectors $(\mathbf{x}, \mathbf{y}) \in \mathbb{R}^n \times \mathbb{R}^m$. In supervised learning, *training* $F_{\boldsymbol{\theta}}$ is the process of finding a good estimate of $\boldsymbol{\theta}$ such that the predicted output $F_{\boldsymbol{\theta}}(\mathbf{x})$ is reasonably close to the actual output $\mathbf{y}$ for the entire data set (e.g., by minimizing the sum of mean squared errors over $\boldsymbol{\theta}$). After training (i.e., $\boldsymbol{\theta}$ is chosen and fixed), *prediction* refers to the process of evaluating $F_{\boldsymbol{\theta}}(\mathbf{x})$ on any arbitrary input vector $\mathbf{x}$ in the hope that $F_{\boldsymbol{\theta}}$ generalizes to the unknown $\mathbf{y}$ corresponding to $\mathbf{x}$.

An MLP $F_{\boldsymbol{\theta}}$ consists of $L \geq 2$ *layers*. Let $\mathcal{L} = \{1, \ldots, L\}$ be the set of layers. Layer 1 is the *input layer* and layer L is the *output layer*. The intermediate layers are *hidden layers*. If there are two or more hidden layers, $F_{\boldsymbol{\theta}}$ is described as *deep*.

Each layer consists of *units*, also known as *neurons*. Let $\mathcal{U}_l = \{1, \ldots, U_l\}$ be the set of units in layer $l \in \mathcal{L}$, where $U_l \geq 1$ is the number of units in layer l. The input layer has n units (i.e., $U_1 = n$) and the output layer has m units (i.e., $U_L = m$). Each unit u in layer l represents a function $f_{l,u}$. The units in the input layer (i.e., $l = 1$) represents the input vector $\mathbf{x}$ to $F_{\boldsymbol{\theta}}$. Formally, unit $u \in \mathcal{U}_1$ represents the identity function $f_{1,u} : \mathbb{R}^n \to \mathbb{R}$ of the uth component of $\mathbf{x} = (x_1, \ldots, x_n)$:

$$f^{1,u}(\mathbf{x}) = x_u.$$

In a *fully-connected* MLP, every unit $u \in \mathcal{U}_l$ in layer $l > 1$ represents a function $f^{l,u}_{\mathbf{W}_{l,u}, B_{l,u}} : \mathbb{R}^{U_{l-1}} \to \mathbb{R}$ with parameters $\mathbf{W}_{l,u} \in \mathbb{R}^{U_{l-1}}$ and $B_{l,u} \in \mathbb{R}$ that maps the outputs of the units in the previous layer $l - 1$ to one real number:

$$f^{l,u}_{\mathbf{W}_{l,u}, B_{l,u}}(\mathbf{x}) = \sigma^{l,u}(\mathbf{W}_{l,u} \cdot \mathbf{x} + B_{l,u}).$$

The vector $\mathbf{W}_{l,u}$ is the *weights* of u, and the scalar $B_{l,u}$ is the *bias* of u. The function $\sigma^{l,u} : \mathbb{R} \to \mathbb{R}$ is the *activation function* of u. Common activation functions include the sigmoid function $\sigma(x) := \frac{1}{1+e^{-x}}$ and the rectified linear unit (ReLU) $\sigma(x) := \max(x, 0)$. In recent times, ReLU has displaced the sigmoid activation function [10]. For this reason, this study focuses on ReLU.

Let $\boldsymbol{\theta} = (\mathbf{W}_{2,1}, B_{2,1}, \ldots, \mathbf{W}_{2,U_2}, B_{2,U_2}, \ldots, \mathbf{W}_{L,1}, B_{L,1}, \ldots, \mathbf{W}_{L,U_L}, B_{L,U_L})$ be the concatenation of the parameters of all units. Then, the MLP $F_{\boldsymbol{\theta}}$ is a function composition of all units through its layers, as described above.

It is well-known that MLPs with at least one hidden layer and a sufficiently large number of units is a universal function approximator under reasonable assumptions (e.g., [7]). In simpler terms, MLPs can approximate any arbitrary transition function given enough units and appropriate activation functions. For this reason, we use an MLP as the transition function.

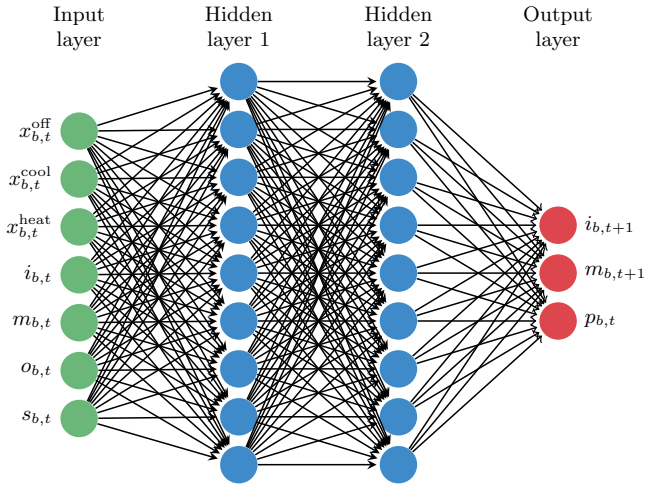

Fig. 2. The architecture of the MLP used to approximate the transition function of each building $b \in \mathcal{B}$.

Let $\mathcal{T} = \{0, \ldots, T\}$ be the time periods and $\mathcal{T}' = \{0, \ldots, T-1\}$ be the planning periods in which decisions are made. Consider a set $\mathcal{B}$ of buildings. Every building $b \in \mathcal{B}$ is associated with an MLP illustrated in Figure 2. It takes the following inputs from the building's environment at timestep $t \in \mathcal{T}'$:

1. An indicator $x^{\text{off}}_{b,t} \in \{0, 1\}$ of whether the HVAC system is off.
2. An indicator $x^{\text{cool}}_{b,t} \in \{0, 1\}$ of whether the HVAC system is cooling.
3. An indicator $x^{\text{heat}}_{b,t} \in \{0, 1\}$ of whether the HVAC system is heating.
4. The indoor air temperature $i_{b,t} \in [0, 1]$ scaled to between 0 and 1.
5. The indoor mass temperature $m_{b,t} \in [0, 1]$ scaled to between 0 and 1.
6. The outdoor air temperature $o_{b,t} \in [0, 1]$ scaled to between 0 and 1.
7. The solar irradiance $s_{b,t} \in [0, 1]$ scaled to between 0 and 1.

These seven inputs are fed through several fully-connected hidden layers with ReLU activation functions. The output layer has three units:

1. The indoor air temperature $i_{b,t+1} \in [0, 1]$ at the next timestep.
2. The indoor mass temperature $m_{b,t+1} \in [0, 1]$ at the next timestep.
3. The power usage $p_{b,t} \in [0, 1]$ for running the mode given in the input layer.

Unusually, the output units also use the ReLU activation function to ensure non-negativity. (We observed minuscule negative power usage otherwise.) All inputs and outputs are normalized to values between 0 and 1 for numerical reasons.

3.3 The Mixed Integer Quadratic Programming Model

MIQP generalizes mixed integer linear programming by allowing quadratic terms in the objective function. The MIQP model of the problem is shown in Figure 3.

The model minimizes power consumption and uses soft constraints to penalize the difference from the indoor air temperatures to a fixed comfortable temperature. The problem contains two classes of (hard) constraints: (1) constraints that encode the state transition functions, and (2) constraints that limit the total power usage during a subset of timesteps for demand response.

This deterministic model assumes that the buildings' environments (i.e., the outdoor air temperature and solar irradiance) and the power supply constraints are known in advance with certainty. These assumptions are acceptable in practice because 24-hour weather forecasts are adequately accurate, and power supply limitations are ordered by utilities in advance because of extreme heat forecasts (e.g., higher than 40°C). Furthermore, since real-time control problems are solved repeatedly throughout the day, large errors (e.g., drifts) are mitigated in the next run when more accurate data is available.

The model declares three primary decision variables $x^{\text{off}}_{b,t}, x^{\text{cool}}_{b,t}, x^{\text{heat}}_{b,t} \in \{0,1\}$ to respectively indicate whether the HVAC system is off, cooling or heating in building $b \in \mathcal{B}$ during time $t \in \mathcal{T}'$. Define a decision variable $p_{b,t} \in [0,1]$ for the power usage due to operating in the mode indicated by $x^{\text{off}}_{b,t}$, $x^{\text{cool}}_{b,t}$ or $x^{\text{heat}}_{b,t}$. Let $i_{b,t}, m_{b,t}, o_{b,t}, s_{b,t} \in [0,1]$ be the indoor air temperature, indoor mass temperature, outdoor air temperature and solar irradiance of building $b \in \mathcal{B}$ in timestep $t \in \mathcal{T}$. As the model is deterministic, the outdoor temperature $o_{b,t}$ and the solar irradiance $s_{b,t}$ for all timesteps, and the initial conditions $i_{b,0}$ and $m_{b,0}$ are known constants. The remaining $i_{b,t}$ and $m_{b,t}$ (i.e., where $t > 0$) are decision variables.

Every building b has a copy of its MLP transition function for every timestep; totaling T copies for every building. Let $\mathcal{L}_b = \{1, \ldots, L_b\}$ denote the layers of the transition function of b, and $\mathcal{U}^l_b = \{1, \ldots, U^l_b\}$ the units in layer $l \in \mathcal{L}_b$. For every $b \in \mathcal{B}$ and $t \in \mathcal{T}'$, denote the output value of unit $u \in \mathcal{U}^l_b$ in layer $l \in \mathcal{L}_b$ as $f^{l,u}_{b,t} \in [0,1]$. Using the transition function, all $f^{l,u}_{b,t}$ (and hence $i_{b,t+1}$, $m_{b,t+1}$ and $p_{b,t}$) are functionally-defined by the seven inputs $x^{\text{off}}_{b,t}$, $x^{\text{cool}}_{b,t}$, $x^{\text{heat}}_{b,t}$, $i_{b,t}$, $m_{b,t}$, $o_{b,t}$ and $s_{b,t}$. Therefore, search is only required on the $x^{\text{off}}_{b,t}$, $x^{\text{cool}}_{b,t}$ and $x^{\text{heat}}_{b,t}$ variables.

Let $c_p \in \mathbb{R}_+$ be the cost of power, scaled appropriately. Define $\mathcal{R} \subset \mathcal{T}$ as the set of timesteps during which the building temperature must be maintained close to an ideal comfortable temperature $i_0 \in [0,1]$. Any deviation away from i_0 is quadratically penalized by a cost $c_i \in \mathbb{R}_+$. Define $\mathcal{Q} \subset \mathcal{T}' \times [0,1]$ as the set of demand response events, which are pairs of a timestep and the amount of power available during that particular timestep.

The first summation in Objective Function (1a) represents the cost of powering the HVAC systems, and the second penalizes deviations from the ideal temperature. Using a quadratic term rather than the absolute value function ensures that a large deviation in one timestep and a small deviation in another timestep is worse than two medium-sized deviations, for example.

Constraint (1b) equates the input layer of the MLPs to the environment of the buildings. Constraint (1c) makes predictions using the MLPs. This constraint is written using the max function but can be linearized using a binary variable [9]. Constraint (1d) equates the output layers to the temperatures in the

$$\min \sum_{b\in\mathcal{B}}\sum_{t\in\mathcal{T}'} c_p p_{b,t} + \sum_{b\in\mathcal{B}}\sum_{t\in\mathcal{R}} c_i(i_{b,t}-i_0)^2 \tag{1a}$$

subject to

$$(f_{b,t}^{1,1},\ldots,f_{b,t}^{1,7}) = (x_{b,t}^{\text{off}}, x_{b,t}^{\text{cool}}, x_{b,t}^{\text{heat}}, i_{b,t}, m_{b,t}, o_{b,t}, s_{b,t}) \qquad \forall b\in\mathcal{B}, t\in\mathcal{T}', \tag{1b}$$

$$f_{b,t}^{l,u} = \max(\mathbf{W}_{b,t}^{l,u}\cdot(f_{b,t}^{l-1,1},\ldots,f_{b,t}^{l-1,U_{l-1}}) + B_{b,t}^{l,u}, 0)$$
$$\forall b\in\mathcal{B}, t\in\mathcal{T}', l\in\mathcal{L}_b\setminus\{1\}, u\in\mathcal{U}_b^l, \tag{1c}$$

$$(i_{b,t+1}, m_{b,t+1}, p_{b,t}) = (f_{b,t}^{L_b,1}, f_{b,t}^{L_b,2}, f_{b,t}^{L_b,3}) \qquad \forall b\in\mathcal{B}, t\in\mathcal{T}', \tag{1d}$$

$$x_{b,t}^{\text{off}} + x_{b,t}^{\text{cool}} + x_{b,t}^{\text{heat}} = 1 \qquad \forall b\in\mathcal{B}, t\in\mathcal{T}', \tag{1e}$$

$$\sum_{b\in\mathcal{B}} p_{b,t} \le q \qquad \forall (t,q)\in\mathcal{Q}, \tag{1f}$$

$$x_{b,t}^{\text{off}}, x_{b,t}^{\text{cool}}, x_{b,t}^{\text{heat}} \in \{0,1\} \qquad \forall b\in\mathcal{B}, t\in\mathcal{T}', \tag{1g}$$

$$i_{b,t}\in[0,1] \qquad \forall b\in\mathcal{B}, t\in\{1,\ldots,T\}, \tag{1h}$$

$$m_{b,t}\in[0,1] \qquad \forall b\in\mathcal{B}, t\in\{1,\ldots,T\}, \tag{1i}$$

$$p_{b,t}\in[0,1] \qquad \forall b\in\mathcal{B}, t\in\mathcal{T}', \tag{1j}$$

$$f_{b,t}^{l,u}\in[0,1] \qquad \forall b\in\mathcal{B}, t\in\mathcal{T}, l\in\mathcal{L}_b, u\in\mathcal{U}_b^l. \tag{1k}$$

Fig. 3. The MIQP model.

next timestep and the power usages in the current timestep. Constraint (1e) enforces exactly one mode for each building and timestep. Constraint (1f) couples the buildings together by limiting the total power consumption across all buildings according to the demand response events. Constraints (1g) to (1k) are the domains of the decision variables. (The implementation omits Constraints (1b) and (1d) and uses the MLP inputs and outputs directly.)

4 Search Heuristics

This section describes the branching rules and the LNS.

4.1 The Variable and Value Selection Strategy

MP solvers and lazy clause generation CP solvers are able to derive good branching rules dynamically by collecting statistics during the search (e.g., [8,1]). However, in classical finite-domain CP solvers, a problem-specific search procedure may be required to be successful. This section presents one such branching rule.

Recall from Section 3.3 that branching is only required on $x_{b,t}^{\text{off}}$, $x_{b,t}^{\text{cool}}$ and $x_{b,t}^{\text{heat}}$ because all other variables are functionally-defined by the operating modes and the input data. The branching rule is driven by one main ideology. During timestep t, the branching rule prefers the operating mode that brings the indoor

temperature nearest to the ideal comfort temperature i_0 if $t+1$ requires temperature control for occupant comfort (i.e., $t+1 \in \mathcal{R}$). Otherwise, the branching rule prefers turning off the HVAC systems to reduce power costs.

The variable and value selection heuristic is described as follows. The algorithm begins by generating a random ordering of the buildings. It then loops through the timesteps from 0 to $T-1$. At every timestep t and building b ordered randomly, it computes the indoor temperature at $t+1$ for all three HVAC operating modes. If $t+1 \notin \mathcal{R}$ (i.e., the temperature in $t+1$ does not need to be controlled for occupant comfort), the branching rule first branches for the HVAC system to be off ($x_{b,t}^{\text{off}} \leftarrow 1$) to reduce power costs, and then branches either cooling ($x_{b,t}^{\text{cool}} \leftarrow 1$) or heating ($x_{b,t}^{\text{heat}} \leftarrow 1$) according to the difference between the ideal comfort temperature i_0 and predicted temperature given cooling or heating. If $t+1 \in \mathcal{R}$, then the predicted temperature after operating in each of the three modes is calculated, and the branching rule branches on the modes in order of difference between i_0 and the predicted temperature. Put simply, if the next timestep does not require temperature control for comfort, the branching rule prefers reduced power costs by branching on turning the HVAC systems off. Otherwise, the branching rule prefers the operating mode that increases comfort. Since calculating these predictions involves many matrix multiplications, the predictions are calculated once during the initialization phase and the branching rule is fixed for the entire search.

4.2 The Large Neighborhood Search

LNS is a popular local search technique [16,19]. It begins with an initial feasible solution, perhaps found using a greedy method. Using this solution, LNS fixes a subset of the variables to their values in the existing solution and then calls an exact solver on the remaining *relaxed* variables. If a better solution is found, it is stored as the incumbent solution. Upon completing the search, the process is repeated with a new search on a different subset of variables fixed to the values in the new best solution. The choice of variables to fix and relax is determined by a subroutine called a *neighborhood*. For LNS to be effective, several neighborhoods should be implemented to target different causes of suboptimality. Six neighborhoods are developed. The six neighborhoods are randomly selected with equal probability. They are described as follows.

Cool Off Neighborhood The Cool Off neighborhood attempts to reduce overcooling of one building. Consider a building operating in a sequence of ⟨Cool, Cool, Cool, Off⟩. Perhaps it is better to have ⟨Off, Cool, Off, Cool⟩ instead. This neighborhood shuffles around the Off and Cool modes within a range of timesteps, and leaves the Heat decisions alone by fixing them in all timesteps. This neighborhood is sketched below:

1. For every building $b \in \mathcal{B}$ and comfort time $t \in \mathcal{R}$, compute a weight $w_{b,t} = (i_{b,t} - i_0)^2$ if $x_{b,t-1}^{\text{cool}} = 1$ and $i_{b,t} < i_0$, and $w_{b,t} = 0$ otherwise. Exit if all $w_{b,t} = 0$.

2. Select a $b^* \in \mathcal{B}$ and $t^* \in \mathcal{R}$ with probability $\frac{w_{b^*,t^*}}{\sum_{b\in\mathcal{B}}\sum_{t\in\mathcal{R}} w_{b,t}}$.
3. Fix $x^{\text{heat}}_{b^*,t} = 0$ for all $t \in \mathcal{T}'$. Relax $x^{\text{off}}_{b^*,t}$ and $x^{\text{cool}}_{b^*,t}$ for all $t \in \{t^* - 1 - k, \ldots, t^* - 1 + k\} \cap \mathcal{T}'$ for some radius $k \in \mathbb{N}$. Fix all other mode variables to their values in the incumbent solution.

Heat Off Neighborhood The Heat Off neighborhood is the heating equivalent of the Cool Off neighborhood:

1. For every building $b \in \mathcal{B}$ and comfort time $t \in \mathcal{R}$, compute a weight $w_{b,t} = (i_{b,t} - i_0)^2$ if $x^{\text{heat}}_{b,t-1} = 1$ and $i_{b,t} > i_0$, and $w_{b,t} = 0$ otherwise. Exit if all $w_{b,t} = 0$.
2. Select a $b^* \in \mathcal{B}$ and $t^* \in \mathcal{R}$ with probability $\frac{w_{b^*,t^*}}{\sum_{b\in\mathcal{B}}\sum_{t\in\mathcal{R}} w_{b,t}}$.
3. Fix $x^{\text{cool}}_{b^*,t} = 0$ for all $t \in \mathcal{T}'$. Relax $x^{\text{off}}_{b^*,t}$ and $x^{\text{heat}}_{b^*,t}$ for all $t \in \{t^* - 1 - k, \ldots, t^* - 1 + k\} \cap \mathcal{T}'$ for some radius $k \in \mathbb{N}$. Fix all other mode variables to their values in the incumbent solution.

Flip Neighborhood The Flip neighborhood attempts to remove sequences of alternating cooling and heating:

1. Create a random ordering of $\mathcal{B}$ and a random ordering of $\mathcal{T}'$.
2. Loop through $b^* \in \mathcal{B}$ and $t^* \in \mathcal{T}'$ using the random orderings. Find a b^* and t^* such that (1) $x^{\text{cool}}_{b^*,t^*} = 1$ and $x^{\text{heat}}_{b^*,t^*+1} = 1$, (2) $x^{\text{heat}}_{b^*,t^*} = 1$ and $x^{\text{cool}}_{b^*,t^*+1} = 1$, (3) $x^{\text{cool}}_{b^*,t^*} = 1$, $x^{\text{off}}_{b^*,t^*+1} = 1$ and $x^{\text{heat}}_{b^*,t^*+2} = 1$ or (4) $x^{\text{heat}}_{b^*,t^*} = 1$, $x^{\text{off}}_{b^*,t^*+1} = 1$ and $x^{\text{cool}}_{b^*,t^*+2} = 1$. Exit if such a b^* and t^* are not found.
3. For some radius $k \in \mathbb{N}$, relax $x^{\text{off}}_{b^*,t}$, $x^{\text{cool}}_{b^*,t}$ and $x^{\text{heat}}_{b^*,t}$ for all $t \in \{t - k, \ldots, t + 1 + k\} \cap \mathcal{T}'$ for cases (1) and (2), and for all $t \in \{t - k, \ldots, t + 2 + k\} \cap \mathcal{T}'$ for cases (3) and (4). Fix all other mode variables to their values in the incumbent solution.

Precool Neighborhood The Precool neighborhood aims to cool down a building before the first timestep of an interval of comfort times:

1. Create a random ordering of $\mathcal{B}$.
2. Loop through $b^* \in \mathcal{B}$ in random order and $t^* \in \mathcal{R}$ in ascending order. Find a b^* and t^* such that $t^* - 1 \notin \mathcal{R}$, $x^{\text{cool}}_{b^*,t^*-1} = 1$ and $i_{b^*,t^*} > i_0$. Exit if such a b^* and t^* are not found.
3. Create an empty set $\tau \subset \mathcal{T}'$ of timesteps.
4. Loop t backwards from $t^* - 2$ to 0. Add t to τ if $x^{\text{cool}}_{b^*,t} = 0$. Stop if $|\tau| = 2k$ for some size parameter $k \in \mathbb{N}$.
5. If $|\tau| = 0$, go back to step (2) to find another b^* and t^*.
6. Fix $x^{\text{heat}}_{b^*,t} = 0$ for all $t \in \mathcal{T}'$. Relax $x^{\text{off}}_{b^*,t}$ and $x^{\text{cool}}_{b^*,t}$ for all $t \in \tau$. Fix all other mode variables to their values in the incumbent solution.

Preheat Neighborhood The PREHEAT neighborhood is the heating equivalent of the PRECOOL neighborhood:

1. Create a random ordering of $\mathcal{B}$.
2. Loop through $b^* \in \mathcal{B}$ in random order and $t^* \in \mathcal{R}$ in ascending order. Find a b^* and t^* such that $t^* - 1 \notin \mathcal{R}$, $x^{\text{heat}}_{b^*,t^*-1} = 1$ and $i_{b^*,t^*} < i_0$. Exit if such a b^* and t^* are not found.
3. Create an empty set $\tau \subset \mathcal{T}'$ of timesteps.
4. Loop t backwards from $t^* - 2$ to 0. Add t to τ if $x^{\text{heat}}_{b^*,t} = 0$. Stop if $|\tau| = 2k$ for some size parameter $k \in \mathbb{N}$.
5. If $|\tau| = 0$, go back to step (2) to find another b^* and t^*.
6. Fix $x^{\text{cool}}_{b^*,t} = 0$ for all $t \in \mathcal{T}'$. Relax $x^{\text{off}}_{b^*,t}$ and $x^{\text{heat}}_{b^*,t}$ for all $t \in \tau$. Fix all other mode variables to their values in the incumbent solution.

On Neighborhood The ON neighborhood attempts to turn on the HVAC systems:

1. Initialize $w^{\text{cool}}_t = 0$ and $w^{\text{heat}}_t = 0$ for all $t \in \mathcal{R}$.
2. For every building $b \in \mathcal{B}$ and comfort time $t \in \mathcal{R}$, add a weight $(i_{b,t} - i_0)^2$ to w^{cool}_t if $x^{\text{cool}}_{b,t-1} = 0$ and $i_{b,t} > i_0$, or add the weight to w^{heat}_t if $x^{\text{heat}}_{b,t-1} = 0$ and $i_{b,t} < i_0$. Exit if all $w^{\text{cool}}_t = 0$ and $w^{\text{heat}}_t = 0$.
3. Focus on cooling with probability $\frac{\sum_{t\in\mathcal{R}} w^{\text{cool}}_t}{\sum_{t\in\mathcal{R}} w^{\text{cool}}_t + w^{\text{heat}}_t}$. Focus on heating otherwise.
4. If cooling, select a $t^* \in \mathcal{R}$ with probability $\frac{w^{\text{cool}}_{t^*}}{\sum_{t\in\mathcal{R}} w^{\text{cool}}_t}$. Relax $x^{\text{off}}_{b,t}$ and $x^{\text{cool}}_{b,t}$ for all $b \in \mathcal{B}$ and $t \in \{t^* - 1 - k, \ldots, t^* - 1 + k\} \cap \mathcal{T}'$ for some radius $k \in \mathbb{N}$.
5. If heating, select a $t^* \in \mathcal{R}$ with probability $\frac{w^{\text{heat}}_{t^*}}{\sum_{t\in\mathcal{R}} w^{\text{heat}}_t}$. Relax $x^{\text{off}}_{b,t}$ and $x^{\text{heat}}_{b,t}$ for all $b \in \mathcal{B}$ and $t \in \{t^* - 1 - k, \ldots, t^* - 1 + k\} \cap \mathcal{T}'$ for some radius $k \in \mathbb{N}$.
6. Fix all other mode variables to their values in the incumbent solution.

5 Experimental Results

This section presents the experimental set-up and analyses the results.

5.1 Generating Exploratory Training Data

To generate the exploratory data for training the neural networks, we assume we have access to physics-based Equivalent Thermal Parameter (ETP; [17]) models that are closely matched to the real-world dynamics. By using models, we avoid having to operate real buildings at uncomfortable temperatures far from their set points for long periods of time needed to gather sufficient data. We instantiate an ETP model for each of twenty buildings based on defaults provided by the grid simulator GridLAB-D [6] plus some small ($\sigma = 0.05\mu$) Gaussian noise to vary their structural parameters.

The training data is generated as follows: we use the ETP models to generate 25,000 two-hour trajectories on five-minute timesteps, resulting in 600,000 data

points per building model. Every sample trajectory consists of a fixed outdoor temperature sampled uniformly at random, $o_0 \sim \mathcal{U}(0, 45)$, an initial indoor mass temperature $m_0 \sim \mathcal{U}(10, 35)$, and corresponding initial air temperature close to the initial mass temperature, $i_0 \sim \mathcal{U}(-0.5, 0.5) + m_0$. Then, a sequence of 24 exploratory actions are taken, by choosing actions uniformly from $\{\textsc{Off}, \textsc{Heat}, \textsc{Cool}\}$, and recording for every step: the initial conditions, the power consumed by the HVAC system, and the indoor air and indoor mass temperatures after five minutes.

5.2 Training the Neural Networks

The training data is preprocessed by scaling all values to lie between 0 and 1. This ensures that the loss in all dimensions are evenly considered. The MLPs are trained using the Adam stochastic gradient descent algorithm [11] over 100 epochs in Tensorflow. One-fifth of the input trajectories are reserved for validation. For every building, ten randomly-initialized training runs are conducted. Of the ten runs, the trained parameters resulting in the smallest sum of mean squared errors are chosen and fixed for planning in the MIQP model.

Three sizes of MLPs are trained: (1) three hidden layers, each consisting of thirty ReLU units, (2) two hidden layers, each with twenty ReLU units and (3) two hidden layers, each with ten ReLU units. Averaged over the twenty buildings, the three sizes respectively have 1.6×10^{-8}, 1.5×10^{-8} and 4.7×10^{-8} validation loss. We use the smallest MLPs since they still accurately fit the data and the largest MLPs have marginally more error than the middle option.

5.3 Experimental Set-up

The MIQP model is solved using Gecode 6.2.0 and Gurobi 9.0.1. Using these two solvers, the following four methods are evaluated:

- *CP-BB*: Branch-and-bound search in Gecode with the branching rule described in Section 4.1.
- *MP-BB*: Branch-and-bound search in Gurobi.
- *CP-LNS*: Local search in Gecode with the branching rule from Section 4.1 and the LNS from Section 4.2.
- *MP-LNS*: Local search in Gurobi with the LNS from Section 4.2.

Both Gecode and Gurobi have built-in support for the max function in the ReLU activation function. Hence, the models are directly implemented, and no attempt was made to manually linearize Constraint (1c) for Gurobi. The radius parameter k for the neighborhoods is randomly chosen as $k = 2$ with probability 0.7, $k = 3$ with probability 0.2 and $k = 4$ with probability 0.1.

All parameters in Gecode are set to their default values. Default parameters are also used in Gurobi except the search is set to prioritize the primal bound and the solver is warm-started with a greedy solution found using a procedure almost identical to the branching rule from Section 4.1.

The solvers are tested on 504 instances. First, 42 typical weather conditions (correlated outdoor temperature and solar irradiance time series) are generated, of which 9 cover cold days with extremely low temperatures, and 9 are hot days with extremely warm temperatures. Then, we control the complexity of the instances generated by varying the length of the planning horizon, and the number of buildings to be controlled. The horizon varies between four, eight and sixteen hours, divided into five-minute time periods resulting in 48, 96, and 192 timesteps, respectively. Each of these instances is then paired with 5, 10, 15 and 20 buildings, making up the 504 instances. Every instance has at least one demand response event, located at the coldest or warmest parts of the day. The initial conditions of each building are given, with initial indoor mass temperature assumed to be equal the initial air temperature. Every instance is run by each of the four methods on a single thread for thirty minutes on an Intel Xeon E5-2660 v3 CPU at 2.60GHz.

5.4 Results and Analysis

Figure 4 is a cactus plot showing for each objective value how many instances have better objective value as found by each solver. The chart shows that LNS outperforms complete branch-and-bound, and that CP-LNS is significantly better than MP-LNS. MP-BB and MP-LNS find feasible solutions to 125 and 215 instances respectively. CP-BB finds solutions to 174 instances, and CP-LNS finds solutions to all 504 instances. Even though both MP-BB and MP-LNS are given a warm start solution, they are unable to activate the solution in many instances, declaring unknown problem status (unknown infeasible or feasible) at termination. Output logs suggest that Gurobi needs to perform computations to activate the solution. This may be caused by the need to compute a basis, which is hindered by the large number of binary variables internally added to linearize the max constraints.

CP-LNS, MP-LNS, CP-BB and MP-BB find the (equally) best solution to 439, 37, 8 and 20 instances respectively. Given the short time-out, neither exact method is able to prove optimality on any instance. The optimality gap averages 97% across the 125 instances with feasible solutions from MP-BB. At best, the optimality gap is 69%. These numbers suggest that the model features a weak linear relaxation. Overall, these results indicate that exact methods are ineffective for real-time control problems that have short time-outs required for repeated reoptimization.

The MIQP model has simple structure, which should make it relatively easy for Gurobi. Even though ReLU functions are nearly linear (i.e., they are a max function composed with an affine function), they actually belong to the class of non-convex programming problems. (Recall that a convex minimization problem has convex less-than-or-equal-to constraints and affine equality constraints [5].) Therefore, we speculate that the linearization of the ReLU function using a binary variable leads to a very weak linear relaxation, making Gurobi inadequate. Rather, the fast tree search of CP and its avoidance of computing nearly useless dual bounds are key to tackling the problem.

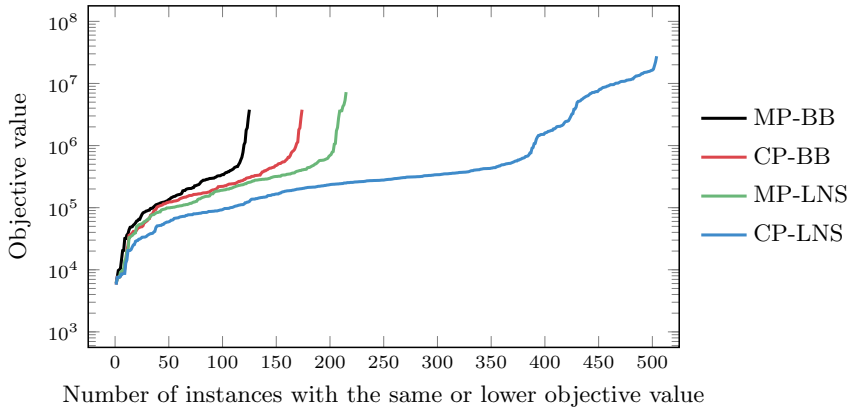

Fig. 4. Cactus plot of objective value against the number of instances for which each method finds better solutions. On the left of any one value on the horizontal axis is the number of instances that the method finds a solution with the same or lower objective value.

5.5 Closing the Loop

The objective values reported in Figure 4 are predicted by the optimization routine using the learned dynamics. To verify that the learned transition models are accurate, we applied the computed schedules to an ETP simulator with the actual parameters. Figure 5 (left) presents the match between the predicted temperature trajectory as optimized by CP-LNS and the ground-truth trajectory obtained by running the schedule through the simulator for an arbitrary instance. We verified that this behavior is consistent across all instances.

Figure 5 (right) compares the accuracy of the learned dynamics with the same optimization routine using a simpler MLP that omits the internal mass temperature. We observe that training on all simulator inputs is required to get schedules that closely match the predicted behavior. We hypothesize that the reason for the large errors observed from the air-only MLP is that this model does not actually constitute a function: for every input of ⟨air, outdoor, solar irradiance, control⟩, there are multiple correct corresponding next air temperatures. The MLP learns to fit a least-error mean value, but this necessarily gives an error in the prediction, which compounds over time into the dramatic mismatch observed here.

6 Conclusions and Future Work

This paper developed an MIQP model for real-time control of space conditioning systems while considering demand response. The problem adjusts the indoor temperature of several smart buildings connected to a microgrid to ensure occupant comfort while subject to sporadic limitations on power supply. The model is solved using the MP solver Gurobi and the CP solver Gecode. The paper

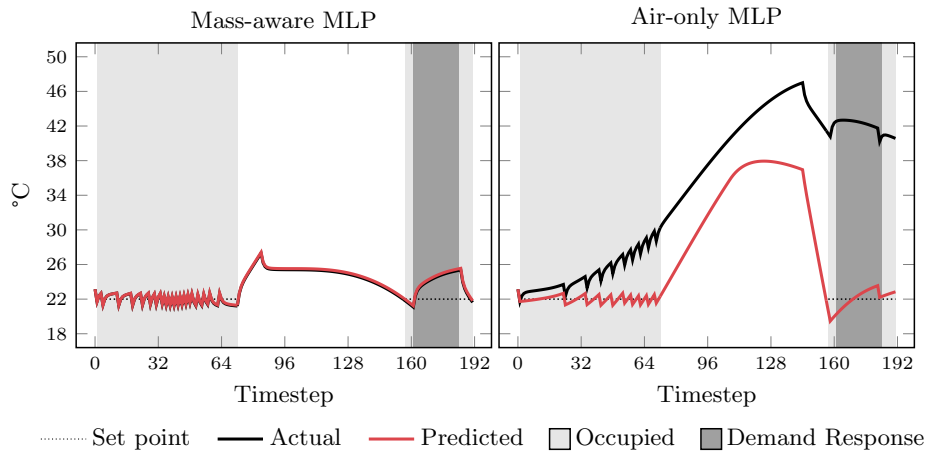

Fig. 5. Comparison of the scheduled and actual temperature trajectory of a building. Left: MLP trained on both air and mass temperatures. Right: MLP trained only on air temperatures.

develops a problem-specific branching rule and LNS, which comprises six neighborhoods that attempt to repair six different reasons for suboptimality. Gecode coupled with the branching rule and LNS vastly outperforms Gurobi.

Several directions for future research are available:

- More sophisticated measures of occupant comfort include various other factors, such as humidity. Future studies should include these variables.
- The model was initially implemented in CP Optimizer but it faced numerical issues. CP solvers with lazy clause generation should be used, but it appears that no other production-ready CP solver supports floating point variables.
- Dedicated propagators for the ReLU function should be investigated. The outputs of the neural network are calculated using a sequence of matrix multiplications, which can be implemented very efficiently using single instruction multiple data (SIMD) CPU instructions. A propagator that reasons over the entire neural network, rather than an individual neuron, could potentially improve the CP solver to a large extent.
- Different kinds of approximations to the transition functions should be evaluated. Deep neural networks are non-convex, making them difficult to reason over. Linear models and decision trees are simpler and using them can potentially improve the solving run-time.
- The control problem is naturally reoptimized over a rolling time window. Adding one additional time period and keeping the existing decisions fixed may not heavily impact the solution quality, but not does not allow a large search space to be explored. This early study investigates a once-off overnight run. Considering that the temperature approximations will drift as the day progresses, the number of timesteps to reoptimize during the day is a key question for future studies.

References

1. Achterberg, T., Koch, T., Martin, A.: Branching rules revisited. Operations Research Letters **33**(1), 42–54 (2005)
2. Anderson, R., Huchette, J., Tjandraatmadja, C., Vielma, J.P.: Strong mixed-integer programming formulations for trained neural networks. In: Lodi, A., Nagarajan, V. (eds.) Integer Programming and Combinatorial Optimization. pp. 27–42. Springer International Publishing, Cham (2019)
3. Azuatalam, D., Mhanna, S., Chapman, A., Verbič, G.: Optimal HVAC scheduling using phase-change material as a demand response resource. In: 2017 IEEE Innovative Smart Grid Technologies-Asia (ISGT-Asia). IEEE (2017)
4. Bartolini, A., Lombardi, M., Milano, M., Benini, L.: Neuron constraints to model complex real-world problems. In: Lee, J. (ed.) Principles and Practice of Constraint Programming. pp. 115–129. Springer Berlin Heidelberg, Berlin, Heidelberg (2011)
5. Boyd, S., Vandenberghe, L.: Convex Optimization. Cambridge University Press (2004)
6. Chassin, D.P., Fuller, J.C., Djilali, N.: GridLAB-D: An agent-based simulation framework for smart grids. Journal of Applied Mathematics **2014** (2014)
7. Cybenko, G.: Approximation by superpositions of a sigmoidal function. Mathematics of Control, Signals and Systems **2**(4), 303–314 (1989)
8. Feydy, T., Stuckey, P.J.: Lazy clause generation reengineered. In: Gent, I.P. (ed.) Proceedings of the 15th International Conference on the Principles and Practice of Constraint Programming. pp. 352–366. Springer Berlin Heidelberg (2009)
9. FICO: MIP formulations and linearizations (2009), https://www.fico.com/en/resource-download-file/3217
10. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT Press (2016), http://www.deeplearningbook.org
11. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: ICLR 2015 (2015)
12. Kohlhepp, P., Harb, H., Wolisz, H., Waczowicz, S., Müller, D., Hagenmeyer, V.: Large-scale grid integration of residential thermal energy storages as demand-side flexibility resource: A review of international field studies. Renewable and Sustainable Energy Reviews **101**, 527–547 (March 2019)
13. Motegi, N., Piette, M.A., Watson, D.S., Kiliccote, S., Xu, P.: Introduction to commercial building control strategies and techniques for demand response. Tech. rep., California Energy Commission, PIER (2006)
14. de Nijs, F., Stuckey, P.J.: Risk-aware conditional replanning for globally constrained multi-agent sequential decision making. In: Proceedings of the 19th International Conference on Autonomous Agents and Multiagent Systems. IFAAMAS (2020)
15. Pérez-Lombard, L., Ortiz, J., Pout, C.: A review on buildings energy consumption information. Energy and Buildings **40**(3), 394–398 (January 2008)
16. Pisinger, D., Røpke, S.: Large neighborhood search. In: Gendreau, M., Potvin, J.Y. (eds.) Handbook of metaheuristics, chap. 13, pp. 399–419. Springer (2010)
17. Pratt, R.G., Taylor, Z.T.: Development and testing of an equivalent thermal parameter model of commercial buildings from time-series end-use data. Tech. rep., Pacific Northwest Laboratory, Richland, Washington (1994)
18. Say, B., Wu, G., Zhou, Y.Q., Sanner, S.: Nonlinear hybrid planning with deep net learned transition models and mixed-integer linear programming. In: Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17. pp. 750–756 (2017)

19. Shaw, P.: Using constraint programming and local search methods to solve vehicle routing problems. In: Maher, M., Puget, J.F. (eds.) Principles and Practice of Constraint Programming, Lecture Notes in Computer Science, vol. 1520, pp. 417–431. Springer Berlin Heidelberg (1998)
20. Vázquez-Canteli, J.R., Nagy, Z.: Reinforcement learning for demand response: A review of algorithms and modeling techniques. Applied Energy **235**, 1072–1089 (2019)