MARLISA: Multi-Agent Reinforcement Learning with Iterative Sequential Action Selection for Load Shaping of Grid-Interactive Connected Buildings

Jose R. Vazquez-Canteli
Department of Civil, Architectural and Environmental Engineering
The University of Texas at Austin
Austin, TX, USA
jose.vazquezcanteli@utexas.edu

Gregor Henze
Dept. of Civil, Environmental and Architectural Engineering
University of Colorado Boulder
Boulder, CO, USA
gregor.henze@colorado.edu

Zoltan Nagy
Department of Civil, Architectural and Environmental Engineering
The University of Texas at Austin
Austin, TX, USA
nagy@utexas.edu

ABSTRACT

We demonstrate that multi-agent reinforcement learning (RL) controllers can cooperate to provide more effective load shaping in a model-free, decentralized, and scalable way with very limited sharing of anonymous information. Rapid urbanization, increasing electrification, the integration of renewable energy resources, and the potential shift towards electric vehicles create new challenges for the planning and control of energy systems in smart cities. Energy storage resources can help better align peaks of renewable energy generation with peaks of electricity consumption and flatten the curve of electricity demand. Model-based controllers, such as MPC, require developing models of the systems controlled, which is often not cost-effective or scalable. Model-free controllers, such as RL, have the potential to provide good control policies cost-effectively and leverage the use of historical data for training. However, it is unclear how RL algorithms can control a multitude of energy systems in a scalable coordinated way. In this paper, we introduce MARLISA, a controller that combines multi-agent RL with our proposed iterative sequential action selection algorithm for load shaping in urban energy systems. This approach uses a reward function with individual and collective goals, and the agents predict their own future electricity consumption and share this information with each other following a leader-follower schema. The RL agents are tested in four groups of nine simulated buildings, with each group located in a different climate. The buildings have diverse load and domestic hot water profiles, PV panels, thermal storage devices, heat pumps, and electric heaters. The agents are evaluated on the average of five normalized metrics: annual net electric consumption, 1 – load factor, average daily peak demand, annual peak demand, and ramping. MARLISA achieves superior results over multiple independent/uncooperative RL agents using the same reward function. Our results outperformed a manually optimized rule-based controller (RBC) benchmark by reducing the average daily peak load by 15%, ramping by 35%, and increasing the load factor by 10%. A multi-year case study on real weather data shows that MARLISA significantly outperforms the RBC in within a year and converges in less than 2 years. Combining MARLISA and the RBC for the first year improves overall initial performance by learning from the RBC rather than random exploration.

ACM ISBN 978-1-4503-8061-4/20/11…$15.00
https://doi.org/10.1145/3408308.3427604

CCS CONCEPTS

• Computing methodologies → Artificial Intelligence → Distributed Artificial Intelligence → Multiagent systems

KEYWORDS

Reinforcement learning, multi-agent coordination, microgrid, demand response

ACM Reference format:

Jose R. Vazquez-Canteli, Gregor Henze, Zoltan Nagy. 2020. MARLISA: Multi-Agent Reinforcement Learning with Iterative Sequential Action Selection for Load Shaping of Grid-Interactive Connected Buildings. In The Proceedings of ACM. (BuildSys '20), November 18–20, 2020*, Yokohama, Japa*n. https://doi.org/10.1145/3408308.3427604

1 Introduction

Buildings account for over 70% of the electricity consumption in the US [1], and the increasing electrification in urban areas, when combined with additional renewable energy resources, can help decrease carbon emissions [2]. Distributed electricity resources have the potential to help buildings match their peaks of electricity consumption with the peaks of renewable energy generation. Demand response can help buildings play an active role in the generation and storage of electricity by coordinating with each other to increase demand flexibility and provide load shaping capabilities. When large amounts of energy storage are available, there is a risk of the energy consuming agents acting greedily by taking non-coordinated peak shifting actions that can lead to an overall peak shifting rather than peak reduction [3]. The anticipated increase of electric vehicles (EVs) and batteries in the near future

will increase the importance of taking well-coordinated control decisions that prevent new peaks of electricity consumption from being formed at times when demand for electricity is traditionally low [4]. Therefore, demand response must also allow the coordination of multiple buildings such that, at the district level, the net energy peaks are not only shifted but shaved.

Model-based control approaches, such as model-predictive control (MPC), often provide good performance if the model is accurate and does not need to be updated much over time. However, due to the close relationship that there exists between urban energy systems (i.e. buildings, batteries, EVs) and human behavior, it is not always possible or cost effective to model these energy systems due to the variability of changing consumption patterns.

Adaptive model-free controlers, such as reinforcement learning (RL), have the potential to overcome these challenges, as they can find good control policies through the interaction with the system controlled and/or by making use of historical data [5][6]. Although the research in RL for energy management of urban energy systems has gained popularity in recent years, a lot of this research has focused on controlling single agents rather than on the coordination of many interdependent agents as a multi-agent system [7]. Recent research in multi-agent RL has focused on scheduling electric water heaters for energy efficiency [8], and various loads in residential buildings [9] without sharing of state information.

In this paper, we introduce a novel algorithm for coordinated demand response: MARLISA, a multi-agent RL algorithm with iterative sequential action selection. MARLISA allows turning any off-policy RL algorithm into a decentralized scalable multi-agent RL algorithm that allows the coordination of its agents and provide demand response capabilities. MARLISA is inspired by the asymmetric multi-agent reinforcement learning algorithm [10], which we have adapted to the energy load shaping problem. This approach uses a reward function with individual and collective goals, and the agents predict their own future electricity consumption and share this information with each other following a leader-follower schema. We demonstrate our algorithm in groups of nine buildings in four climate zones, and compare it to a reference rule-based controller (RBC), as well as an RL algorithm with independent agents (without coordination). We evaluate the controllers for their ability to reduce peak demand, net electricity consumption, ramping and maximize the load factor [11].

The rest of this paper is structured as follows. First, an overview about RL and the soft-actor critic algorithm. Then, we introduce MARLISA, and show how it can turn common off-policy single-agent RL into multi-agent decentralized RL. We then discuss the reward design, and the simulation environment CityLearn. Sections 3 and 4 show and discuss the results. Section 5 concludes the paper.

1.1 Reinforcement learning

Reinforcement learning (RL) is an agent-based and potentially model-free control algorithm that learns through interaction with the environment that it is controlling [12] (see Figure 1). The objective of the agent (controller) is to maximize the expected

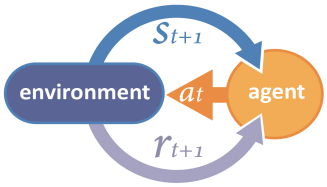

Figure 1 Agent-environment interaction in RL

cumulative sum of the discounted rewards over time. RL is formalized using a Markov Decision Process (MDP), which contain sets of states S, actions A, a reward function $R: S \times A$ and transition probabilities between the states, $P: S \times A \times S \in [0,1]$. The policy π represents a mapping between states and actions, $\pi: S \rightarrow A$, and the value function $V^{\pi}(s)$ is the expected return for the agent starting in state s and following the policy π, i.e.,

$$V^{\pi}(s) = \sum_{a} \pi(s,a) \sum_{s'} P^{a}_{s\,s'}[\,R^{a}_{s\,s'} + \gamma V^{\pi}(s')] \qquad (1)$$

where $R^{a}_{s\,s'}$, which can be denoted as $r(s,a)$, is the reward obtained after taking an action $a = \pi(s_k)$, and transitioning from the current state s to the next state s', and $\gamma \in [0,1]$ is a discount factor for future rewards. An agent that uses $\gamma = 1$ will consider future rewards as important as current rewards, whereas for $\gamma = 0$, greater values are assigned to states that lead to high immediate rewards.

In model-free RL, the environment's dynamics (transition probabilities P) are unknown. Q-learning is probably the most widely known model-free RL technique [13]. For tasks with small state sets, the transitions can be represented with a table that stores the state–action values (Q-values). Each entry in the table is a state–action tuple (s,a), and the Q-values are updated as

$$Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha[r(s_t,a_t) + \gamma \max_{a} Q(s_{t+1},a_t) - Q(s_t,a_t)] \qquad (2)$$

where s' is the next state, and $\alpha \in (0,1)$ is the learning rate, which explicitly defines to what degree new knowledge overrides old knowledge. Q-values represent the expected cumulative sum of the discounted rewards after taking an action under a certain state and following a greedy policy thereafter. Q-learning is also an off-policy method, which can perform updates to its policy using past experience, which could have been collected using different policies. The collected experience, or state-action-reward tuples, are stored in a memory replay buffer and sampled from it to perform the updates iteratively using (2).

1.2 Soft-actor critic reinforcement learning

To control environments that have continuous states and actions, tabular Q-learning is not practical, as it suffers the curse of dimensionality. Actor-critic RL methods use artificial neural networks to generalize across the state-action space: The actor network maps the current states to the actions that it estimates to be optimal. Then, the critic network evaluates those actions by mapping them, together with the states under which they were taken, to the Q-values.

Soft actor-critic (SAC) is a model-free off-policy RL algorithm [14]. As an off-policy method, SAC can reuse experience and learn from fewer samples. SAC is based on three key elements: an actor-critic architecture, off-policy updates, and entropy maximization for efficient exploration and stable training. SAC learns three different functions: the actor (policy), the critic (soft Q-function), and the value function V, defined as

$$\begin{aligned} V(\mathrm{s_t}) &= \mathbb{E}_{\mathrm{a_t}\sim\pi_\theta}[Q(\mathrm{s_t},\mathrm{a_t}) - \alpha \log \pi_\theta(\mathrm{a_t}|\mathrm{s_t})] \quad (3)\\ &= \mathbb{E}_{a_t\sim\pi_\theta}[Q(s_t,a_t)] + \alpha\mathbb{E}_{a_t\sim\pi_\theta}[log\, \pi_\theta(a_t|s_t)]\\ &= \mathbb{E}_{\mathrm{a_t}\sim\pi_\theta}[Q(\mathrm{s_t},\mathrm{a_t})] + \alpha H \end{aligned}$$

where $H \geq 0$ is the Shannon entropy of the policy π_θ, which, under state $\mathrm{s_t}$, is the probability distribution of the possible actions that the agent can take. The policy with zero entropy is the deterministic policy ($\pi_\theta = 0$ for all actions except for the optimal action $\mathrm{a_t^*}$, where $\pi_\theta(\mathrm{a_t^*}|\mathrm{s_t}) = 1$). Policies with non-zero entropies allow more randomized action selection. The objective of the SAC agent is to learn the optimal stochastic policy π^*, given by

$$\pi^* = \arg\max_{\pi_\theta} \sum_{t=0}^{T} \mathbb{E}_{(s_t,a_t)\sim\rho_\pi}\left[r(s_t,a_t) + \alpha H\big(\pi_\theta(\cdot\,|s_t)\big)\right] \quad (4)$$

The final optimal policy can be made deterministic by simply selecting the expected action of the policy (mean of the distribution) as the action of choice, which is what we do in order to evaluate our agents after training. Where $(s_t, a_t)_{\sim\rho_\pi}$is a state-action pair sampled from the agent's policy, $r(s_t, a_t)$ is the reward for a given state-action pair. Due to the added entropy term, the agent will attempt to maximize the returns while behaving as randomly as possible. The parameters of the critic networks are updated by minimizing the expected error J_Q between the predicted Q-values and the ones calculated through iteration, i.e.,

$$J_Q = \mathbb{E}_{(S_t,a_t)\sim D}\left[\frac{1}{2}\Big(Q_\theta(s_t,a_t) - \big(r(s_t,a_t) + \gamma\mathbb{E}_{\mathrm{s_{t+1}}\sim p}[V_{\bar{\theta}}(s_{t+1})]\big)\Big)^2\right] \quad (5)$$

For more details about SAC we refer the reader to [15].

In (3) and (4) $\alpha \in (0,1)$ is a hyperparameter called the temperature, which must not be confused with the learning rate in (2). The temperature controls the importance of the entropy term, and therefore, the stochasticity of the learned: $\alpha = 1$ would prioritize behaving as stochastically as possible, which might lead to uniformly random behavior, while $\alpha = 0$ would ignore entropy altogether and focus the agent on maximizing return without exploration, leading to an almost deterministic policy. In this work we set the temperature to a constant value $\alpha = 0.2$.

2 Methodology

We implement MARLISA in CityLearn, a simulation platform for the implementation of RL for demand response. In the following sections, we review CityLearn briefly and introduce MARLISA in Section 2.2, which also contains a description of how the reward function was designed and the type of action selection algorithm we used. The following subsections of the methodology describe the reference rule-based controller (RBC) used for comparison, and the multi-year case study used for evaluation. Notice that the results show in Section 3.3 correspond to the methodology described in Section 2.4 (multi-year case study), while the results from Section 3.1 and 3.2 correspond to the simulation of the single-year datasets in four different climate zones, which are described in Section 2.1. We use the terms "district", "micro-grid", and "group" interchangeably. Similarly, "building" and "agent" are used interchangeably.

2.1 CityLearn

CityLearn is an OpenAI Gym environment for the easy implementation of RL agents in a multi-agent demand response setting to reshape the aggregated curve of electrical demand by controlling the energy storage of a diverse set of buildings [16]. Its main objective is to facilitate and standardize the evaluation of RL agents such that it enables benchmarking of different algorithms. CityLearn includes energy models of air-to-water heat pumps, electric heaters, and the pre-computed energy loads of the buildings, which include space cooling, dehumidification, appliances, domestic hot water (DHW), and solar generation. We implemented MARLISA into CityLearn to control the storage of DHW and chilled water. The RL agents send their control actions hourly and receive a set of states and rewards in return. Indoor temperatures in the buildings do not change, as the environment automatically constraints the actions of the controllers and ensures that the energy supply devices are large enough to guarantee that the energy demand of the buildings is always supplied. The RL agents can decide how much cooling or heating energy store or release at any given time. A backup controller integrated in CityLearn guarantees that the energy supply devices prioritize satisfying the energy demand of the building before storing any additional energy.

We use four datasets of 9 buildings each, consisting of the DOE prototype buildings [17]. The energy demand for each building has been pre-simulated using EnergyPlus in a different climatic zone of the USA (2A | Hot-Humid | New Orleans; 3A | Warm-Humid | Atlanta; 4A | Mixed-Humid | Nashville; 5A | Cold-Humid | Chicago) [18]. The group of buildings that we use in this research is composed of one medium office (id=1), one fast-food restaurant (id=2), one standalone retail (id=3), one strip mall retail (id=4), and five medium multi-family buildings (id=5-9). Their respective cooling and DHW storage capacities, as well as PV installed capacities, are shown on Table 1. Cooling and DHW storage capacities are represented as the multiple of hours the storage device can satisfy the maximum annual hourly cooling or DHW demand if fully charged.

Each building has an air-to-water heat pump and most buildings also have an electric heater that supplies them with DHW. All these

devices, together with other electric equipment and appliances (non-shiftable loads) consume electricity from the main grid. Photovoltaic generation can offset part of this electricity consumption by allowing the buildings generate their own electricity.

Table 1 Sizing of the energy supply and storage equipment of the different buildings (See text for details)

	Cooling Storage				DHW Storage [h]				PV [kW]			
id	2A	3A	4A	5A	2A	3A	4A	5A	2A	3A	4A	5A
1	3	2	1.5	4	3	2	1.5	4	120	0	20	0
2	3	3	2	3	3	2.5	3	3	0	30	20	0
3	3	2	3	3	0	0	0	0	0	0	0	0
4	3	2.5	2	2	3	0	0	0	40	0	40	25
5	3	1.5	2	2	3	2	3	2	25	25	15	0
6	3	3	3	3	3	3	3	3	20	25	5	0
7	3	3	1.5	3	3	3	2	3	0	25	0	0
8	3	3.5	3	2	3	3	3	3	0	25	0	20
9	3	3	3	3	3	3	3	2	0	25	0	0

To create instances of the non-shiftable electrical loads of the residential buildings (id=5-9), we used Pecan Street data [19] from multiple households in Austin, TX, and trained probabilistic regression models and generated dozens of electricity profiles of appliances to feed the EnergyPlus models with. We used the same approach for the domestic hot water profiles using open-source data from the Solar Row project [20]. For more realistic energy consumption profiles, we generated diverse cooling and heating temperature setpoints for the different thermal zones of the multi-family buildings using data from the ResStock project [21].

2.2 MARLISA

MARLISA is an extension of SAC that allows for coordination of the agents through reward sharing, as well as mutual sharing of information. To coordinate, each agent only needs to share two variables with one other agent, which makes our algorithm scalable, as the number of variables needed by each agent does not increase with the number of agents. MARLISA can be applied to decentralized multi-agent problems, such as coordinated load shaping in groups of buildings.

The implementation of MARLISA in CityLearn is shown in Figure 2. Each building in the district has its own RL agent. Our objective is to design these agents such that they learn to coordinate with each other when initialized with a random policy and without knowledge of the system dynamics. From an energy perspective, the agents should learn to shape the overall load of the micro-grid. Specifically, we evaluate the agents in their ability to minimize the annual peak net demand, the average daily peak, the total ramping, the annual net demand, and maximize the average daily load factor of the entire district [11].

We used the SAC algorithm we discussed in section 1.2, which we modified for the multi-agent implementation. We used a learning rate $lr = 3e^{-4}$, a decay rate $\tau = 5e^{-3}$, $batch\ size = 256$, $\gamma_{discount} = 0.99$, and $256x256$ for all neural network

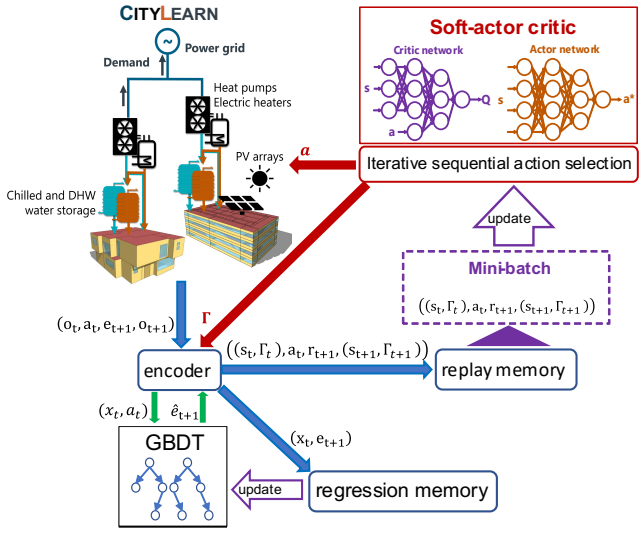

Figure 2 Simulation framework: integration of MARLISA into CityLearn

architectures (size of hidden layers). We used the Huber loss [22], as this loss is less affected by outliers, and we applied layer normalization to all the layers of the critic networks, as this has been proven to speed up the training [23]. We also applied principal component analysis to the encoded observations and reduced their number of dimensions by 25%. We implemented our agents in PyTorch and we will release the implementations with this paper on Github [24].

2.2.1 Reward Design. We experiment with different reward structures, as shown in Table 2. The rewards $r_i^1, r_i^2, and\ r_i^3$ are all *single-agent* rewards, as their value only depends on the net electricity consumption e_i of building/agent i. $e_i < 0$ if the building is consuming more electricity that it generates, and $e_i > 0$ if the building is self-sufficient at that time and generates excess electricity. r_i^{MARL} is a combination of the individual net electricity use e_i and the collective component $\sum e_i$, i.e., the total net electricity consumption of the entire district, and is used to share information between the agents, which rewards them for reducing the coordinated energy demand. We found empirically that an exponent 2 for the e_i and an exponent of 1 for the collective factor performed well.

Table 2 Reward functions compared in this research

r_i^1	$\boldsymbol{min\{0, e_i\}}$
r_i^2	$\boldsymbol{sign(e_i) \cdot min\{0, e_i\}^2}$
r_i^3	$\boldsymbol{min\{0, e_i\}^3}$
r_i^{MARL}	$\boldsymbol{-sign(e_i) \cdot e_i^2 \cdot \min\left\{0, \sum_{i=0}^{n} e_i\right\}}$

At the beginning of the simulation, during the random exploration phase of two epochs, we collect all the rewards and calculate their mean and standard deviation (STD). Then, we

normalize all the collected rewards and any future rewards by subtracting from them

the calculated mean and dividing by the STD. Normalizing the rewards using constant values for mean and STD is necessary, as using different values recalculated during the simulation could invalidate the Q-values that the agents had already learned. Once the rewards are normalized, we scale them by multiplying them by a factor of 5, as recommended in [15]. This normalization of the rewards ensures that the RL agents can be tuned with similar hyperparameters. For instance, the selection of an optimal learning rate and temperature is sensitive to the values of the rewards.

Table 3 shows how the sign of r_i^{MARL} changes depending on the values of its individual and collective factors. The reward only becomes negative if the building is consuming electricity from the grid while the district is also consuming electricity from the main grid. The reward becomes positive when the building generates more electricity than it consumes while the district is consuming electricity from the main grid, since the building is contributing to making the district self-sufficient energetically. If the district is self-sufficient (no electricity consumption from the main grid), all the agents receive a reward equal to 0.

Table 3 Reward domain for different values of the agent's net electricity consumption and the district's net electricity demand. e_i>0: the building generates more than it consumes

r_i^{MARL}	$e_i > 0$	$e_i < 0$	$e_i = 0$
$\sum_{i=0}^{n} e_i \geq 0$	0	0	0
$\sum_{i=0}^{n} e_i < 0$	+	−	0

2.2.2 Sequential iterative action selection with shared predictions. The use of collective rewards as described above provides each agent with a more accurate description of the actual goal of the district. However, it also increases the stochasticity of the reward, as the states of each agent cannot explain the changes in the collective factor of the reward. Our approach to remediate this is to share certain information between the agents that provides each agent the information they need to make a more accurate prediction of what the next reward will be under the current state if they take a certain action. The reward r_i^{MARL} that each agent receives does not only depend on its individual net electricity consumption but also on the total net electricity demand of the entire district. Therefore, we train a gradient boosting decision tree (GBDT) for each building, using a normalized subset of the observations o_i, and the actions of the agent, to predict the net electricity consumption of the building on the next time step if a certain action is taken (See Figure 3).

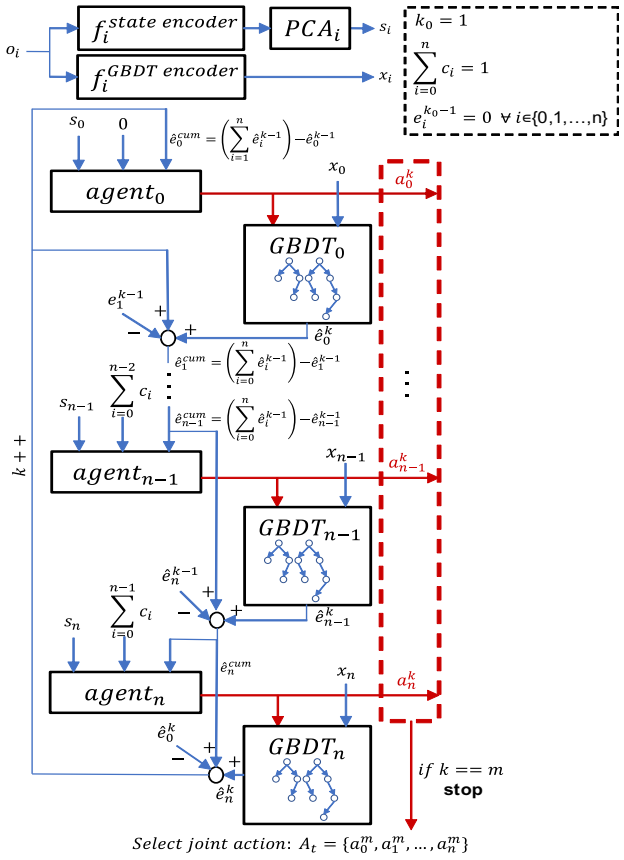

Figure 3 Sequential iterative action selection and shared predictions of MARLISA

In brief, every time that the action selection function is called, a randomly sorted list of all agents, is created. The first agent on the list picks an action (a_i^k), its GBDT model makes a prediction, $\hat{e}_i^k$, of how much electricity the building will consume if that action is taken (e_i^k), and this information is shared with the next agent. Each new agent that selects an action shares with the following agent the cumulative predicted electricity consumption of itself and the agents that selected an action before it ($\hat{e}_i^{cum}$). At this stage, the actions are selected, but not yet taken, until all the agents have selected their actions and shared their predictions with each other in the same order m times. Then, the joint action, **A**, is taken (the set of the actions of all the agents).

Note that at the last iteration, m, the first agent selecting an action does not know what the rest of the agents will do because it only has information from the previous iteration, *m-1*, while the last agent to select an action has complete information. The agents closer to the beginning of the action selection queue have less complete information, which can impact how accurately they can predict future expected rewards. To correct this, we provide them with a coordination variable c_i^{cum}. At the beginning of the simulation, each building is assigned an energy coefficient c_i, which is a rough estimate of their annual electricity consumption

compared to the estimated annual electricity consumption of the district, i.e., the relative weight that the building's energy use has in the micro-grid system. Then, c_{i-1}^{cum} is the sum of all the energy coefficients of the buildings which have selected an action before the current building i in the current iteration k. Therefore, c_i^{cum} will be 0 for the first agent in the random queue, and closer to 1 for the last agent selecting an action. c_i^{cum} indicates each building how much potential energy consumption has already been allocated by the buildings selecting an action before it. $\hat{e}_i^{cum}$ indicates the agent i how much energy is expected to be consumed by all the other buildings (not only the ones before it). Therefore, c_i^{cum} provides each agent with information about how accurate $\hat{e}_i^{cum}$ will be. If, in the last iteration $k = m$, $c_i^{cum} \approx 1$ then $agent_i$ will expect its $\hat{e}_i^{cum}$ to be very accurate, as it will have the actual expected electricity consumption of the whole district. If $c_i^{cum} = 0$, $agent_i$ will be the first of the queue, and will expect $\hat{e}_i^{cum}$ to be less accurate, as it will have been calculated using the predictions of the agents from the previous iteration, k-1.

The algorithm from Figure 3 is formally shown in Algorithm 1. S_{ext} is the extended state, which is made of the RL states s_i defined in Table 4, to which we stack the coordination variables Γ.

Algorithm 1: Action selection of MARLISA

Initialize: $k \leftarrow 1$, $\hat{e}_i^{k=0} \leftarrow 0 \; \forall \, i \in B = \{0, 1, \dots, n\}$
$\hat{e}^{cum} \leftarrow 0$
$s_i \leftarrow f_i^{\text{state encoder}}(o_i)$, $x_i \leftarrow f_i^{\text{GBDT encoder}}(o_i) \; \forall \, i \in B$
while $k \leq m$ **do**
  $i \leftarrow 0$; $c^{cum} \leftarrow 0$
  for all agents $i \in B$ **do**
    $S_{ext} \leftarrow (s_i, c^{cum}, \hat{e}^{cum})$
    $a_i^k \leftarrow SAC_i\_ActionSelection(S_{ext})$
    $\hat{e}_i^k \leftarrow GBDT_i\_Predict(x_i, a_i^k)$
    $c_{i+1}^{cum} \leftarrow c_i^{cum} + c_i$
    if $i < n - 1$ **then**
      $\hat{e}_{i+1}^{cum} \leftarrow \hat{e}_i^{cum} + \hat{e}_i^k - \hat{e}_{i+1}^{k-1}$
    else
      if $i < m$ **then**
        $\hat{e}_0^{cum} \leftarrow \hat{e}_i^{cum} + \hat{e}_i^k - \hat{e}_0^{k-1}$
      else
        $A \leftarrow (a_0^k, a_1^k, \dots, a_n^k)$
        $\Gamma \leftarrow$
  $k \leftarrow k + 1$
return A, Γ

2.2.3 Integration of the RL agents into the environment. The implementation of the agents differs from a traditional RL implementation in that the coordination variables Γ are added to the rest of the states to perform the updates. Therefore, Γ_{t+1} must be calculated before the update, which involves calling the action selection function after the environment moves to the next time step as Algorithm 2 shows.

In Algorithm 2, the environment returns the next state, i.e., a set of observations that we encode using various methods (see Table 4) and transform using principal component analysis (PCA) to obtain the actual states the RL agents see (see the encoder block in Figure 2). Table 4 contains the observed variables that we used to create the states and what methods we used to transform them. We encoded the variables using periodic, one-hot normalization and they were also standardized by subtracting their mean and dividing by their standard deviation. To estimate their mean and standard deviation, we ran the agents for a year following a random policy to collect the samples, as we did with the rewards.

Algorithm 2: integration of MARLISA into the

$s_t \leftarrow$ env.reset()
$A_t, \; \Gamma_t \leftarrow$ **agent.select_action**(s_t)
while done not True **do**
  $s_{t+1}, r_t,$ **done** $\leftarrow$ env.step(A_t)
  $A_{t+1}, \; \Gamma_{t+1} \leftarrow$ **agents.select_action**(s_{t+1})
  agents.update$((s_t, \Gamma_t), a_t, r_t, (s_{t+1}, \Gamma_{t+1}))$
  $s_t \leftarrow s_{t+1}$
  $a_t \leftarrow a_{t+1}$
  $\Gamma_t \leftarrow \Gamma_{t+1}$

Table 4 Transformation of the observed variables into states and predictive variables to be used by the RL agents and the gradient boosting decision trees, respectively.

Observed variable	RL: $o \rightarrow s$	GBDT: $o \rightarrow x$
Hour of day	Periodic	Periodic
Day of week	One-hot	Periodic
Month	Periodic	Periodic
Outdoor temp.	Standardization	Standardization
Pred. Outdoor temp. 6h ahead	Standardization	Remove variable
Pred. Outdoor temp. 12h ahead	Standardization	Remove variable
Pred. Outdoor temp. 24h ahead	Standardization	Remove variable
Outdoor RH	Standardization	Standardization
Pred. Outdoor RH 6h ahead	Standardization	Remove variable
Pred. Outdoor RH 12h ahead	Standardization	Remove variable
Pred. Outdoor RH 24h ahead	Standardization	Remove variable
Indoor temp.	Standardization	Standardization
Indoor RH	Standardization	Standardization
Non-shiftable load	Standardization	Standardization
Cooling storage SoC	Standardization	Standardization
DHW storage SoC	Standardization	Standardization
If the building has solar generation		
Direct solar rad.	Standardization	Standardization
Pred. direct solar rad. 6h ahead	Standardization	Remove variable
Pred. direct solar rad. 12h ahead	Standardization	Remove variable
Pred. direct solar rad. 24h ahead	Standardization	Remove variable
Diffuse solar rad.	Standardization	Standardization
Pred. diffuse solar rad. 6h ahead	Standardization	Remove variable
Pred. diffuse solar rad. 12h ahead	Standardization	Remove variable
Pred. diffuse solar rad. 24h ahead	Standardization	Remove variable
Solar generation	Standardization	Standardization

2.3 Reference rule-based controller

As a baseline to measure performance, we tuned a rule based controller (RBC) to act greedily in every building and use its storage capacity to reduce its energy consumption by storing more energy during the night (when the coefficient of performance of the

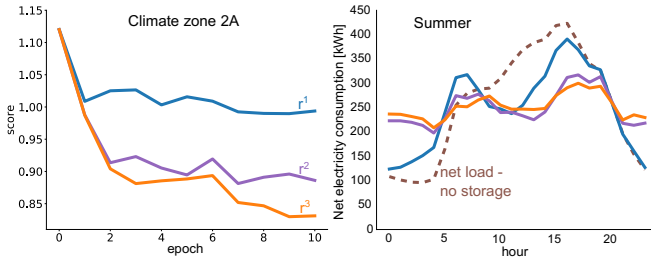

Figure 4 Effect of the different non-collective rewards on the normalized scores of the RL agents (left) and on the median daily net load profile during the Summer (right)

heat pumps is higher) and release it during the day. We assumed no detailed knowledge of the energy profile of each individual building. The RBC was tuned to charge 9.1% of its maximum capacity every hour between 10pm and 8am, and discharge 8% of its maximum capacity every hour between 9am and 9pm.

2.4 Multi-year case study with real weather data

The methodology of the results of subsections 3.1 and 3.2 analyzes the design, convergence, and performance of different types of single-agent and multi-agent RL algorithms. The different simulations are run on 10 epochs using typical mean year weather data (TMY), i.e., every epoch is a repetition of the exact same year. Although this approach is useful to compare the different RL agents with each other, it does not give very insightful information about the practicality of implementing RL in a real-world environment. Therefore, we developed a multi-year case study in which we evaluate the RL controllers on the same set of nine buildings in the climate zone 2A (New Orleans) that we have used throughout this paper. In this case, the RL agents need to learn on-the-go performing exploration-exploitation for a period of 5 years (2005, 2014-2017). The year 2005 was used instead of 2013 because of limited data availability for 2013 and to use a longer time span. With this case study we aimed at answering three main questions:

1. How fast can the RL agent improve its performance, without any prior information, after it is implemented?
2. How bad is the control policy during the exploration period?
3. What methods can be used to increase the learning speed?

Our approach is model-free and does not require any previous model development, although we did assume that there is an inner control loop in the building that guarantees that the storage controller (the RL agent) cannot take any action that leaves the temperature setpoint of the building unmet. Therefore, during the exploration period, the RL agents follow a constrained pseudo-random policy, which has some operational limits imposed to guarantee that the energy demand of the building is always satisfied and that additional energy storage can be attained only when this building energy demand is met.

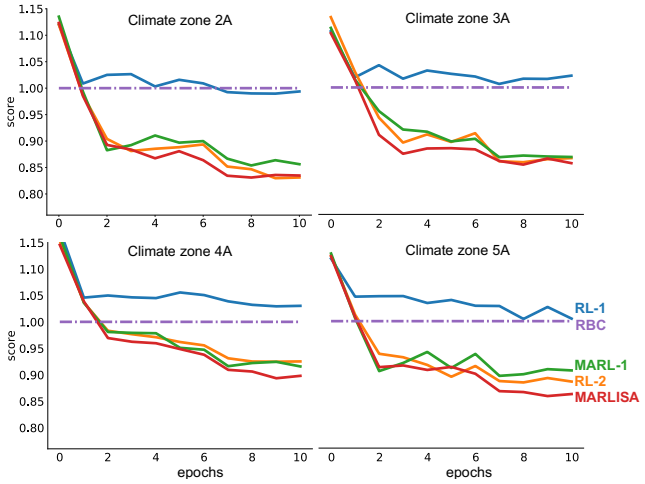

Figure 5 Scores of the RL controllers for all climate zones in the 9-building district. (See text for description of each controller)

MARLISA performed constrained random action exploration for the first 250 days of the simulation. Then, it performed an exploration-exploitation process using SAC to maximize the expected rewards and the entropy of the policy (for 300 more days). Finally, after 550 days into the simulation, MARLISA started to evaluate the stochastic policy deterministically (by choosing the mean value of the policy rather than sampling from it).

We also tested a mixed controller, which used the RBC during the first 250 days of the simulation instead of the random exploration. While the RBC takes the actions, the RL agents collect data from the states, actions, rewards, and coordination variables, and use MARLISA thereafter (RBC + MARLISA in Section 3).

3 Results

3.1 Single-agent reward design

Figure 4 shows the performance of rewards r^1, r^2 and r^3 in climate zone 2A for the district of nine buildings (note that 1, 2, and 3 are superscripts of the rewards, not exponents). Increasing the exponent of the net electricity consumption in the rewards significantly improves the performance of the RL agents by flattening the overall curve of demand more aggressively. No improvement was found for exponents greater than 3.

3.2 Multi-agent RL performance

In Figure 5, we compare the single agents that use r^1 and r^3 (RL-1 and RL-2 in the figure respectively), with the agents using the partially collective reward r^{MARL} (MARL-1 and MARLISA in the figure respectively). MARL-1, as RL-1 and RL-2, does not use sequential iterative action selection algorithm, whereas MARLISA does use it. Despite the fact that both MARL-1 and MARLISA use the same reward function, MARL-1 performs worse than

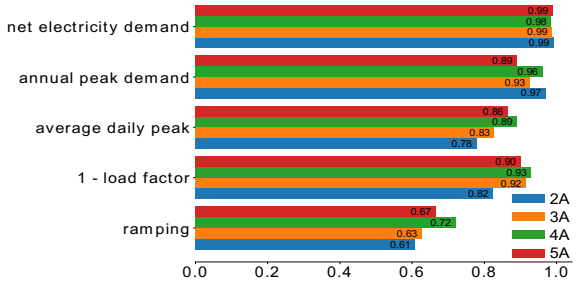

Figure 6 Individual scores of MARLISA for the different climate zones after training and normalized by the scores of our baseline rule-based controller (RBC)

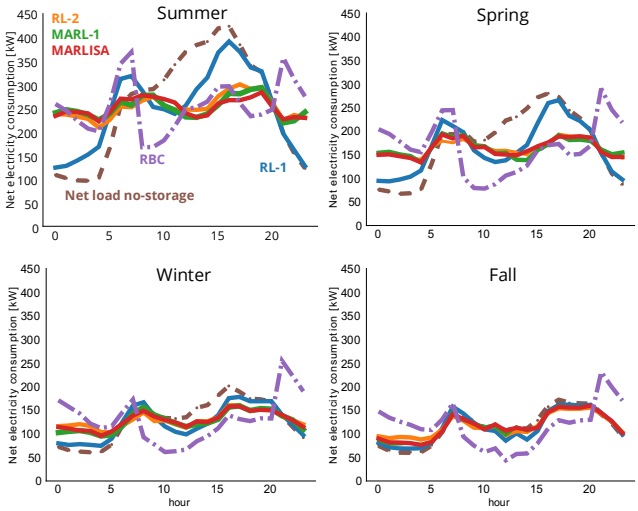

Figure 7 Median daily net load of the district in climate zone 2A for nine buildings

MARLISA, even worse than the single-agent RL algorithm RL-2. The addition of the sequential iterative action selection algorithm makes MARLISA perform better and learn faster than all the other algorithms. RBC is the reference manually optimized rule-based controller to which all the results are normalized.

Figure 6 illustrates the individual scores for the four climate zones analyzed of MARLISA. The major improvement with respect to the baseline RBC happened for the ramping metric (roughly 35% better than our RBC), which demonstrates the ability to flatten the demand curve. The algorithm performed well at reducing the 1 - load factor metric and the average daily peaks (by roughly 10-20%). The net electricity demand remain at the same level than that of the RBC, which is a good result, as the RL reward is aimed at flattening the net load and not particularly at reducing the total amount of electricity consumed. Finally, MARLISA reduced the annual peak demand by about 3-11%, which can probably be improved significantly more.

Figure 7 depicts the median day net load profile for every season of climate zone 2A. All the RL controllers, except RL-1, outperform the RBC. The RL controllers learn a unique control policy that is specific for each building and its associated electricity consumption patterns. Furthermore, MARLISA, also provides

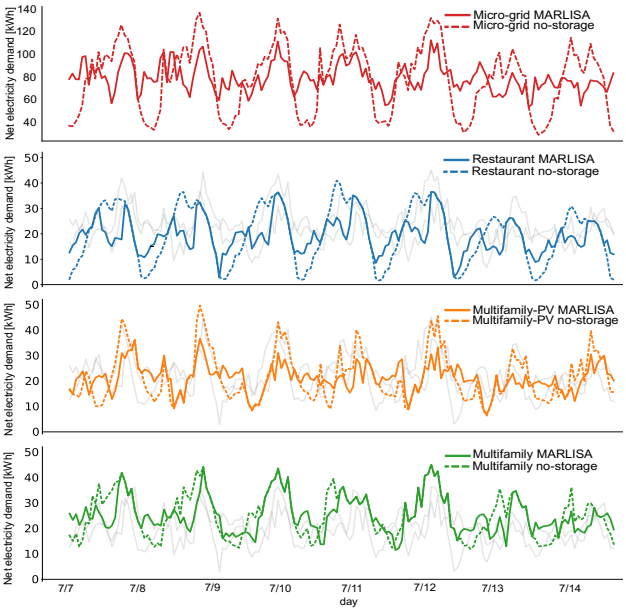

Figure 8 Individual behavior of the agents cooperating to achieve load shaping

some additional coordination among buildings. The RBC, on the other hand, leads each building to act greedily and shifts the peak of net electricity consumption towards the night-time (when the greedy agents expect the district's net load to be lower) instad of shaving it.

For illustration purposes of the coordination dynamics between the buildings, we ran MARLISA for a different system with only 3 buildings: id={2, 5, 7} in climate zone 2A. Figure 8 illustrates how, for a full week in the summer, the controllers make different storage decisions adapting to the different energy profiles of their respective buildings, one of which has a solar panel. The overall net load of the 3-building system is shaved.

3.3 Case study: simulating a real-world implementation

As Figure 9 illustrates, all the RL controllers analyzed outperformed the reference greedy RBC in every metric analyzed except the net electricity consumption, for which all performed similarly. All the subplots in Figure 9 show moving 30-day averages, except the annual peak demand, which shows a one-year moving average. The average score and the annual peak demand metrics are shown starting at the end of year one since both include a trailing one-year moving average. Each algorithm was run 5 times, and we display the median score and the minimum and maximum scores are represented as error bands. The results are consistent with those of Figure 5: MARLISA outperforms the single-agent RL controller (RL-2) in all metrics but the net electricity consumption. After convergence, the RL-2 achieves an average score of about 0.85, and MARLISA of about 0.82. However, during the random exploration period during the first 250 days of the simulation, RL-2 and MARLISA perform much worse than the RBC, which is particularly noticeable during the Summer of year 1. To tackle this problem, we evaluated the controller RBC

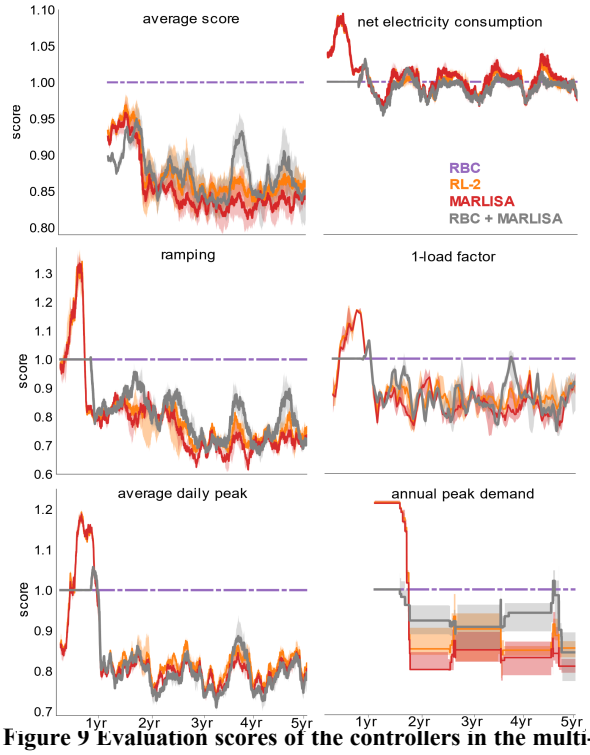

Figure 9 Evaluation scores of the controllers in the multi-year case study. Each controller was run 5 times, the error bands indicate minimum and maximum values and the solid line is the median value

+ MARLISA, which uses the RBC instead of the random exploration period. This controller performed well during this initial phase, but at the expense of some long-term performance, which on average was worse than that of RL-2 and MARLISA.

As Figure 9 shows, the reference RBC performs significantly better load shaping during the Summer and Spring (when load shaping is crucial) than during the Winter (when load shaping is not as relevant), which leads to higher scores of the RL controllers during the Spring and Summer, and lower in Winter. This explains why some scores such as the load factor and the average daily peak start low at the beginning of year one, even though the RL agents are exploring. The figure shows how after less than a year the RL controllers can already coordinate better than the reference RBC, and they converge after approximately 1.5 to 2 years.

Figure 10 depicts the net electricity consumption of the multi-agent controllers for a given week in Summer of year 1 and year 5. As the figure shows, both controllers perform better than the RBC during year 5, but MARLISA performed significantly worse at the beginning, when it is still exploring constrained random actions.

4 Discussion

Our results show how, RL, can be implemented successfully for load shaping in groups of interconnected buildings without knowledge of the system dynamics or need for model development.

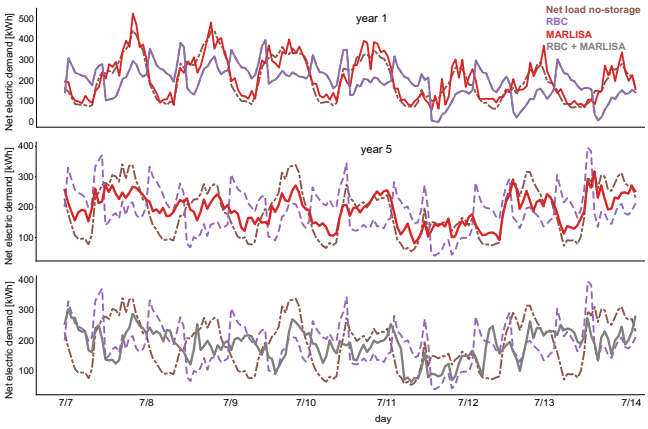

Figure 10 Net electric demand of the district in the case study in the Summer period of year 1 (top), and year 5 (bottom two)

We emphasize that proper reward design is critical. We observed that raising the net electricity consumption to the power of 3 provides the agents with the right single-agent reward to perform aggressive load shaping. The reason is that this reward penalizes high electricity consumption proportionally much more than lower values of electricity consumption, flattening the curve of demand as a result. This design of the reward does not only incentivize flattening the curve of demand but also reducing energy consumption overall while correcting RL inherent bias towards maximizing short term rewards: As RL agents maximize the expected discounted sum of future rewards, the discount factor assigns a higher importance to energy savings that happen immediately than those that happen later. Therefore, RL is inherently biased towards consuming less energy now at the expense of consuming more later. As a hypothetical example consider an agent with a discount factor $\boldsymbol{\gamma = 0.99}$ controlling a building with a heat pump and a lossless thermal storage device.

The agent would not consume and store energy immediately in order to consume it 12 hours later unless the heat pump coefficient of performance (COP) 12 hours later is at least $\left(\mathbf{1} - \frac{\mathbf{1}}{\mathbf{0.99^{12}}}\right)\boldsymbol{x100\% = 12.8\%}$ higher than it is in the present time. This constitutes a problem since, ideally, the agent should store energy now even if the COP were marginally higher 12 hours later. Raising the net electricity demand to a power greater than 1 magnifies any benefit induced by the COP as seen by the agent, which reduces the RL bias towards immediate energy savings.

Adding a collective term to the reward function is a way to share information and promote cooperation to achieve a common goal rather than acting greedily. However, collective rewards violate the Markov property, as the agent does not have state information about the other agents and therefore cannot predict and learn what they will do and what the reward will be. The use of a collective term in the reward function ($\boldsymbol{r^{marl}}$) led to worse results when used by the single-agent RL controller and compared with the use of the

reward r^3. To correct the violation of the Markov property, we combined the partially collective reward function with our iterative sequential action selection method with shared predictions (MARLISA), the agents outperformed all other agents we tested, and the RBC by about 15% (average of our five metrics).

Standardizing the rewards by subtracting their mean and dividing by their standard deviation improved the results and allowed us to use the same hyper parameters across all the agents. Hyper-parameters such as the learning rate or the temperature of the SAC agents are sensitive to the values of the rewards. It is important to use constant values of the mean and the standard deviation (or some estimates of them) rather than recalculating them during the simulation, since this would make already learned Q-values invalid as the new rewards would be standardized differently. Other techniques we used that improved the performance of the agents were applying PCA to decorrelate the states of the agents and reduce their dimensions, layer normalization of the critic networks to speed up training, and use of the Huber loss to avoid exploding gradients and oversensitivity to outliers in the critic networks.

The multi-year case study showed how, without a significant drop in long-term performance, it is possible for the RL agents to collect data from an RBC rather than performing random exploration at the beginning of the learning period. This mitigates one of the problems of RL, which is having a poor initial performance. On the other hand, the other RL agents we tested used a constrained random exploration, that performed poorly in the exploration phase but did not violate any temperature setpoints as it always ensured that the electricity demand of the buildings was satisfied (thanks to properly sizing the energy supply equipment).

Even though MARLISA collects data from the environment to train not only the RL algorithm, but also an internal regression "model", MARLISA is still considered a model-free RL algorithm because no model of the environment dynamics, i.e., transition probabilities are used or learned.

5 Conclusion

The expected increase in storage capacity and electrical load volatility due to the integration of electric vehicles and other additional distributed energy resources such as photovoltaic panels or batteries can make model-based unpractical or very costly. In this paper, we have introduced a novel multi-agent RL implementation, MARLISA, for load shaping in micro-grids. MARLISA uses a leader-follower iterative sequential action selection schema and a reward with a collective component. The proposed algorithm remains scalable regardless of the number of buildings and shares limited information anonymously between the buildings for improved coordination. Our results showed an average improvement of about 15% with respect the load shaping metrics of the baseline rule-based controller, and better results than several single-agent RL controllers we tested. We evaluated the controllers in a 5-year case study that showed how the RL controllers can significantly outperform a typical RBC in within a year, and end convergence in less than 2 years. While rule-based controllers act greedily and uncoordinated, multi-agent RL controllers can cooperate to provide more effective load shaping in a model-free, decentralized, and scalable way with very limited sharing of information in an anonymous way. Thus, offsetting load volatility in micro-grids which is caused by multiple factors such as human behavior, weather conditions, or HVAC control sequences. Future work will focus on improving MARLISA and expanding our open-source simulation platform, CityLearn, to add battery models and the capability to simulate controlled charging and discharging of electric vehicles.

REFERENCES

[1] Society AP. Buildings. Energy Futur. Think Effic., 2008, p. 52–85.
[2] Leibowicz BD, Lanham CM, Brozynski MT, Vázquez-Canteli JR, Castillo N, Nagy Z. Optimal decarbonization pathways for urban residential building energy services. Appl Energy 2018;230:1311–25.
[3] Gelazanskas L, Gamage KAA. Demand side management in smart grid: A review and proposals for future direction. Sustain Cities Soc 2014;11:22–30.
[4] Windham A, Treado S. A review of multi-agent systems concepts and research related to building HVAC control. Sci Technol Built Environ 2016;22:50–66.
[5] Vázquez-Canteli J, Kämpf J, Nagy Z. Balancing comfort and energy consumption of a heat pump using batch reinforcement learning with fitted Q-iteration. Energy Procedia 2017;122:415–20.
[6] Vázquez-Canteli JR, Ulyanin S, Kämpf J, Nagy Z. Fusing TensorFlow with building energy simulation for intelligent energy management in smart cities. Sustain Cities Soc 2019;45:243–57.
[7] Vázquez-Canteli JR, Nagy Z. Reinforcement learning for demand response : A review of algorithms and modeling techniques. Appl Energy 2019;235:1072–89.
[8] Kazmi H, Suykens J, Balint A, Driesen J. Multi-agent reinforcement learning for modeling and control of thermostatically controlled loads. Appl Energy 2019;238:1022–35.
[9] Ahrarinouri M, Rastegar M, Seifi AR. Multi-Agent Reinforcement Learning for Energy Management in Residential Buildings. IEEE Trans Ind Informatics 2020;3203:1–1.
[10] Kononen V. Asymmetric multiagent reinforcement learning. Proc - IEEE/WIC Int Conf Intell Agent Technol IAT'03 2003;2:336–42.
[11] Patteeuw D, Henze GP, Helsen L. Comparison of load shifting incentives for low-energy buildings with heat pumps to attain grid flexibility benefits. Appl Energy 2016;167:80–92.
[12] Sutton R, Barto A. Reinforcement Learning: An Introduction. MIT Press Cambridge, Massachusetts 1998.
[13] Watkins CJCH, Dayan P. Technical Note: Q-Learning. 1992;8:279–92.
[14] Entropy OM, Reinforcement D. Soft Actor-Critic : Off-Policy Maximum Entropy Deep Reinforcement 2016.
[15] Haarnoja T, Zhu H, Tucker G, Abbeel P. [BatchRL-SAC] Soft Actor-Critic Algorithms and Applications 2015.
[16] Vázquez-Canteli JR, Kämpf JH, Henze GH, Nagy ZN. CityLearn v1.0: An OpenAI Gym Environment for Demand Response with Deep Reinforcement Learning. BuildSys '19 Proc 6th ACM Int Conf Syst Energy-Efficient Build Cities, Transp 2019:356–7.
[17] Laboratory PNN. Commercial Prototype Building Models n.d.
[18] Baechler MC, Gilbride TL, Cole PC, Hefty MG, Ruiz K. Guide to Determining Climate Regions by County. Pacific Northwest Natl Lab Oak Ridge Natl Lab 2015;7:1–34.
[19] Street P. Pecan Street Dataport n.d. https://www.pecanstreet.org/dataport/.
[20] Paul Norton, Bob Hendron EH. Building America Field Test Report. Boulder, CO: 2008.
[21] NREL. ResStock. April 17, 2019 n.d. https://github.com/NREL/OpenStudio-BuildStock.
[22] PyTorch. Smooth L1 Loss n.d. https://pytorch.org/docs/master/generated
[23] Ba JL, Kiros JR, Hinton GE. Layer Normalization(Update of Batch Normalization) 2015.
[24] Vázquez-Canteli JR, Kämpf J, Henze GP, Nagy Z. CityLearn - GitHub repository 2019. https://github.com/intelligent-environments-lab/CityLearn.git.