

On-line Building Energy Optimization using Deep Reinforcement Learning

Elena Mocanu, Decebal Constantin Mocanu, Phuong H. Nguyen, Antonio Liotta, Michael E. Webber, Madeleine Gibescu, J.G. Slootweg

Abstract—Unprecedented high volumes of data are becoming available with the growth of the advanced metering infrastructure. These are expected to benefit planning and operation of the future power system, and to help the customers transition from a passive to an active role. In this paper, we explore for the first time in the smart grid context the benefits of using Deep Reinforcement Learning, a hybrid type of methods that combines Reinforcement Learning with Deep Learning, to perform on-line optimization of schedules for building energy management systems. The learning procedure was explored using two methods, Deep Q-learning and Deep Policy Gradient, both of them being extended to perform multiple actions simultaneously. The proposed approach was validated on the large-scale Pecan Street Inc. database. This highly-dimensional database includes information about photovoltaic power generation, electric vehicles as well as buildings appliances. Moreover, these on-line energy scheduling strategies could be used to provide real-time feedback to consumers to encourage more efficient use of electricity.

Index Terms—Deep Reinforcement Learning, Demand Response, Deep Neural Networks, Smart Grid, Strategic Optimization

I. INTRODUCTION

THERE is an energy transition underway since the start of the millennium, comprised primarily of a push towards replacing large, fossil-fuel plants with renewable and distributed generation. It results in increased uncertainty and complexity in both the business transactions and in the physical flows of electricity in the smart grid. Because the built environment is the largest user of electricity, a deeper look at building energy consumption holds a promise for improving energy efficiency and sustainability. Understanding such individual consumption behavior based on the knowledge transfer from the fusion of extensive data collected from the Advanced Metering Infrastructure (AMI) is an essential step to optimize building energy consumption and consequently the effects of its use.

This work is motivated by the hypothesis that an optimal resource allocation of end-user patterns based on daily smart electrical device profiles could be used to smoothly reconcile differences in future energy consumption patterns and the

supply of variable sources such as wind and solar [1]–[3]. It is expected that a cost minimization problem could be solved to activate real-time price responsive behavior [4]. A wide-range of methods have been proposed to solve the building energy and cost optimization problems, including linear and dynamic programming, heuristic methods such as Particle Swarm Optimization (PSO), game theory, fuzzy methods and so on [1]–[7]. Therein, both centralized and decentralized solutions exist, but they fail to consider on-line solutions for large-scale, real databases [6]. More concretely, any time when an optimization is needed, these methods have to compute completely or partially all the possible solutions and to choose the best one. This procedure is time consuming. In the big data era, more and more machine learning methods appear to be suitable to overcome this limitation by automatically extracting, controlling and optimizing the electrical patterns. This can be done by performing successive transformation of the historical data to learn powerful machine learning models to cope with the high uncertainty of the electrical patterns. Then, these models will be capable of generalization and they could be exploited in an on-line manner (i.e. few milliseconds) to minimize the cost or the energy consumption in newly encountered situations. Among all these machine learning models, the ones belonging to the Reinforcement Learning (RL) area are the most suitable for the cost minimization problem, as they are capable to learn an optimal behavior, while the global optimum is not known.

Thus, in the remaining of this paper we focus on RL methods, such as Q-learning [8], and their latest developments. The building environment is modeled using a Markov Decision Process [9] and it can be used to find the best long-term strategies. Prior studies showed that RL methods are able to solve stochastic optimal control problems [10] in the power system area as well as an energy consumption scheduling problem [11] with dynamic pricing [12]. A batch reinforcement learning method was introduced in [13] to schedule a cluster of domestic electric water heaters, and further on applied for smart home energy management [14]. Owing to the curse of dimensionality, these methods fail for large-scale problems. More recently, there has been a revival of interest in combining deep learning with reinforcement learning. Lately, in 2015, an application of Q-learning to deep learning has been successful at playing Atari2600 games at expert human levels [15]. In 2016, another one has defeated for the first time in history the world champion at the game of Go. Complementary with our work, Francois-Lavet et al. has proposed the use of Deep Q-learning for storage scheduling

E. Mocanu, D.C.Mocanu, P.H.Nguyen, A.Liotta, M.Gibescu and J.G. Slootweg are with the Department of Electrical Engineering, Eindhoven University of Technology, Eindhoven, 5600 MB, The Netherlands.(e-mail:{e.mocanu; d.c.mocanu; p.nguyen.hong; a.liotta; m.gibescu; j.g.slootweg}@tue.nl)

Michael E. Webber is with the Department of Mechanical Engineering, The University of Texas at Austin, Austin, TX 78712-1591, USA. (e-mail: webber@mail.utexas.edu)

in microgrids [16]. The above methods represent the starting point of a new research area, known as Deep Reinforcement Learning (DRL), which has evolved through the intersection of reinforcement learning and neural networks. At the same time, in our previous work, we showed that Reinforcement Learning using Deep Belief Networks for continuous states estimation can successfully perform unsupervised energy prediction [17].

Our contribution: In this paper, inspired by the above research developments, we propose for the first time the use of the Deep Policy Gradient method, as part of Deep Reinforcement Learning algorithms, in the large-scale physical context of smart grid - smart building, as follows.

- We propose a new way to adapt DRL algorithms to the smart grid context, with the aim of conceiving a fast algorithm to learn the electrical patterns and to optimize on-line either the building energy consumption or the cost.
- We investigate two DRL algorithms, namely Deep Q-learning (DQN) [15] and Deep Policy Gradient (DPG).
- DPG in its current form is capable to take just one action at a specific time. As in the building context multiple actions have to be taken at the same moment, we propose a novel method to enhance DQN with the capability of handling multiple actions simultaneously.

We evaluate our proposed methods on the PecanStreet database at both the building and aggregated level. In the end, we prove that our proposed methods are able to efficiently cope with the inherent uncertainty and variability in the generation of renewable energy, as well as in the peoples' behavior related with their use of electricity (i.e. charging of electric vehicles). Specifically, we show that the enhanced DPG is more appropriate to solve peak reductions and cost minimization problems than DQN.

The remaining of this paper is organized as follows. Section II describes the problem formulation and Section III we introduce the background and preliminary concepts. Section IV describes our proposed method followed by implementation details in Section V. Results and discussions are provided in Section VI. Finally, we conclude with some directions for future research.

II. PROBLEM FORMULATION

In this context, we aim to reduce load peaks as well as to minimize the cost of energy. Let \mathcal{B} denote the set of buildings, such that $B_i \in \mathcal{B}, \forall i \in \mathbb{N}$ representing the index of the building analyzed. The total building energy consumption E_i is a sum over all power generation P^+ and consumption in a specific interval of time Δt . Therein, based on the shifting capabilities of appliances present in a building we differentiate between flexible power P_d^- , e.g. electric devices $d \in \{1, \dots, m_i\}$, and fixed consumption P^- .

a) Cost minimization problem: In this paper, we assume two price components over the space of \mathcal{B} , such that λ_t^- is the price value set by the utility company for the time-slot t and λ_t^+ represents the price value at which the utility company buys energy from end-users at time-slot t . Therefore,

the optimal cost associated with customer i at time t for an optimization time horizon T can be calculated as

$$\min \sum_{t=1}^T (\lambda_t^+ \sum_{i=1}^n P_{i,t}^+ - \lambda_t^- \sum_{i=1}^n (P_{i,t}^- + \sum_{d=1}^{m_i} a_{i,d,t} P_{i,d,t}^-)) \quad (1)$$

$$\text{s.t.} \quad \sum_{t=1}^T P_i^- \Delta t = E_i, \quad \forall i \in \mathbb{N}, \forall t \in \mathbb{N}, \quad (2)$$

$$\sum_{t=1}^T P_d^- \Delta t = E_d, \quad \forall d \in \mathbb{N}, \forall t \in \mathbb{N}, \quad (3)$$

$$a_{i,d,t} = \{1, 0\}, \quad \forall a \in \mathcal{A}, \forall i \in \mathbb{N}, \forall d \in \mathbb{N}, \forall t \in \mathbb{N}, \quad (4)$$

$$P_{i,t}^+, P_{i,t}^-, P_{i,d,t}^- \geq 0, \quad \forall t = [1 : T] \in \mathbb{N}, \quad (5)$$

$$\lambda_t^+, \lambda_t^- \geq 0, \quad \forall t = [1 : T] \in \mathbb{N}. \quad (6)$$

where $a_{i,d,t} = 1$ if the electrical device is *on* at that specific moment in time, and 0 otherwise. Please note that, in our proposed method, computing $a_{i,d,t}$ is equivalent with the estimation of the actions (see Fig.1).

b) Peak reduction problem: In the special case of constant price, for electricity generation and consumption, with $\lambda_t^+ = \lambda_t^-$, the cost minimization problem becomes a peak reduction problem, defined as

$$\min \sum_{t=1}^T \left(\sum_{i=1}^n P_{i,t}^+ - \sum_{i=1}^n (P_{i,t}^- + \sum_{d=1}^{m_i} a_{i,d,t} P_{i,d,t}^-) \right) \quad (7)$$

Consequently, the constraints following Eq. 1 will remain valid for both problems. However, based on the differences between different types of electrical devices the full range of constraints is larger as explained in the next sections.

c) Electrical device constrains: We are assuming three types of consumption profiles. Firstly, we consider the *time-scaling load*. In respect to this we confine our analysis to the air conditioning load (d_{AC}), as a representative part of a larger set of electrical devices in every building which could be switched on-off for a limited number of times during an optimization horizon, e.g. lights, television, refrigerator. Prior studies show that short-term air conditioning curtailments have a negligible effect on end-user comfort [18]. Secondly, we include the *time-shifting load*, also called deferrable load, that must consume a minimum amount of power over a given time interval. Therein, we model the dishwasher (d_{DW}) as an uninterruptible load, which requires a number of consecutive time steps. Finally the electric vehicle (d_{EV}) was considered as both a *time scaling and shifting load*. A more rigorous formulation of the building electrical components and their associated constrains could be found in [19]. In our case, a complementary probabilistic perspective over the time dependent devices constrains $a_{d,t}$ give us the following assumptions:

A 1: For all d , with P_d^- time-scaling loads, there $\exists \delta_d \in \mathbb{R}_+$ constants over the optimization horizon such that

$$\begin{cases} \sum_t P_d^- \leq \delta_d & \text{if } p(P_d^- = 0|t) \in (0, 1) \\ \sum_t P_d^- = \delta_d & \text{if } p(P_d^- = 0|t) = 0 \end{cases} \quad (8)$$

where $p(P_d^- = 0|t)$ is the probability of the electrical device d to be active at any moment in time t , for all $t = [1 : T] \in \mathbb{N}$.

A 2: For all d , with P_d^- time-shifting loads, there $\exists \delta_d$ constants such that $\sum_t P_d = \delta_d$, for all $t = [1 : T] \in \mathbb{N}$.

Observation 1: In this paper, P^+ (e.g. PV generation) is considered a non-curtailable resource.

Observation 2: All electrical vehicles, d_{EV} , and their associated consumption P_d^- , were considered as time scaling and shifting loads working under the conditions imposed by Assumption 1 and 2.

III. BACKGROUND AND PRELIMINARIES

In this section, we provide a brief overview of reinforcement learning, Markov decision formalism, and deep neural networks.

A. Reinforcement Learning

In a Reinforcement Learning (RL) [9] context, an agent learns to act using a (Partial Observable) Markov Decision Process (MDP) formalism. MDPs are defined by a 4-tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}(\cdot, \cdot), \mathcal{R}(\cdot, \cdot) \rangle$, where:

- \mathcal{S} is the state space, $\forall s \in \mathcal{S}$,
- \mathcal{A} is the action space, $\forall a \in \mathcal{A}$,
- $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$ is the transition function given by the probability that by choosing action a in state s at time t , the system will arrive at state s' at time $t + 1$, such that $p_a(s, s') = p(s_{t+1} = s' | s_t = s, a_t = a)$, and
- $\mathcal{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \mathbb{R}$ is the reward function, where $\mathcal{R}_a(s, s')$ is the immediate reward received by the agent after it performs the transition to state s' from state s .

The agent aims to optimize a stochastic policy $\pi : \mathcal{S} \times \mathcal{A} \times \mathcal{R} \rightarrow \mathbb{R}_+$. Under structure assumption of the environment (i.e. finite states and actions) the Markov decision problem is typically solved using dynamic programming. However, in our built environment, the model has a large (continuous) states space. Therein, the state space is given by the building energy consumption and price at every moment in time, while the action space is highly dependent on the *electric device constrains*. The success of every action a is measured by a reward r . Learning to act in an environment will make the agent to choose actions to maximize future rewards. The value function $Q^\pi(s, a)$ is an expected total reward in state s using action a under a policy π . Currently, one of the most popular reinforcement learning algorithm is Q-learning [8].

B. Deep Neural Networks

The topology of a Deep Neural Network (DNN) architecture is based on multiple layers of neurons. In general, a neuron is a non-linear transformation of the linear sum of its inputs. The first layer models directly the data. A hidden layer in the neural network architecture is build as an array of neurons taking the inputs from the previous layer. The activation function of a neuron on top of k stacked layers in the architecture is using composite functions, such as $x \otimes h_1 \otimes h_2 \otimes \dots \otimes h_k$.

In 2011, it was shown that supervised training of a very deep neural network with hard non-linearities is faster if the hidden layers are composed of Rectified Linear Units (ReLU) [20]. Recently, the logistic sigmoid and the hyperbolic tangent activation are outperformed by ReLU [15], [21], [22]. Formally, ReLU is defined as a function $f(x_i) = \max(0, x_i)$,

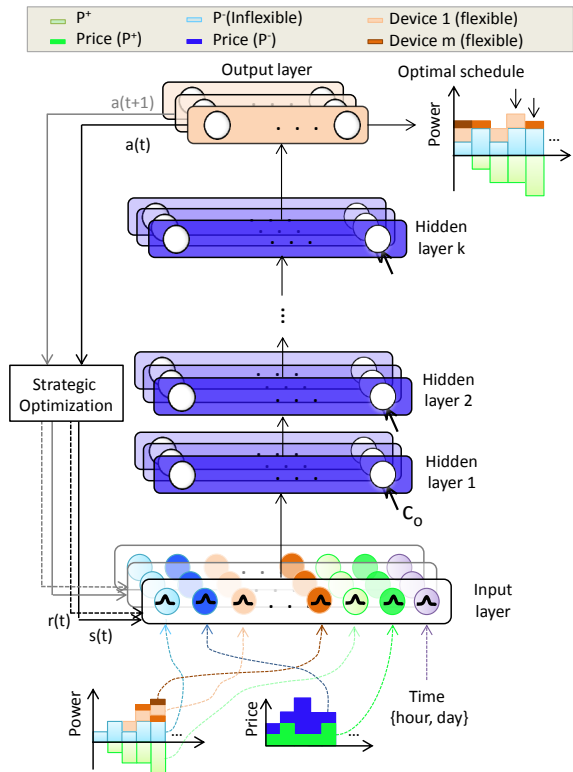


Fig. 1. The closed loop general architecture of Deep Reinforcement Learning, built as a combination of Reinforcement Learning and Deep Neural Network.

where x_i is its input. However, to avoid a non-zero gradient when the hidden units are not active, we used a slightly relaxed form proposed in [23], given by

$$f(x_i) = \begin{cases} x_i & \text{if } x_i > 0, \forall i \in \mathbb{N} \\ \eta x_i & \text{if } x_i \leq 0, \forall i \in \mathbb{N} \end{cases} \quad (9)$$

where η is a coefficient controlling the slope of the negative part. If $\eta = 0$ then Eq. 9 becomes ReLU. One special case of using a DNN is in deep reinforcement learning where the input is given by the states of an MDP and the output represents the actions of the MDP.

IV. PROPOSED METHOD

In this section, we propose the use of Deep Reinforcement Learning (DRL) as an on-line method to perform optimal building resource allocation at different levels of aggregation. The general architecture of our proposed method is depicted in Fig. 1. DRL (RL combined with DNNs of k hidden layers) can learn to act better than the standard RL by automatically extracting patterns, such as those of electricity consumption. Overall, we can represent the DNN method, from a very general perspective, as a black box model with good generalization capabilities over a given input distribution as follows:

$$\frac{\text{Input}}{\text{data}} \rightarrow DNN_{(k)} \xrightarrow{\text{Output}} \text{Data estimation} \quad (10)$$

In the remaining of this section we will introduce two DRL methods, namely Deep Q-learning (DQN) and Deep Policy Gradient (DPG).

$$DRL = \begin{cases} \begin{array}{l} \text{Input} \\ \text{states} \end{array} \rightarrow DNN_{(k)} \begin{array}{l} \text{Output} \\ Q(s,a) \end{array} \rightarrow \text{Deep Q-learning} \\ \begin{array}{l} \text{Input} \\ \text{states} \end{array} \rightarrow DNN_{(k)} \begin{array}{l} \text{Output} \\ p(a|s) \end{array} \rightarrow \text{Deep Policy Gradient} \end{cases}$$

In contrast to value-based methods (e.g. DQN), policy-based model free methods (e.g. DPG) directly parameterize the policy $\pi(a|s; \theta)$ and update the parameters θ by performing, typically approximate, gradient ascent on the expected long-term reward [24].

A. Deep Q-learning (DQN)

Learning in DRL is done as follows. The DNN is trained with a variant of the Q-learning algorithm, using stochastic gradient descent to update its parameters [15]. Firstly, the value-function from the standard RL algorithm is replaced by a deep Q-network with parameters θ , given by the weights and biases of DNN, such that $Q(s, a, \theta) \approx Q^\pi(s, a)$. This approximation is used further to define the objective function by mean-squared error in Q-values

$$\mathcal{L}(\theta) = \mathbb{E}[(r + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}, \theta) - Q(s_t, a_t, \theta))^2] \quad (11)$$

Leading to the following Q-learning gradient

$$\frac{\partial \mathcal{L}(\theta)}{\partial \theta} = \mathbb{E} \left[\left(r + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}, \theta) - Q(s_t, a_t, \theta) \right) \frac{\partial Q(s_t, a_t, \theta)}{\partial \theta} \right] \quad (12)$$

Usually, this standard Q-learning algorithm used in synergy with neural networks oscillates or diverges, mainly because data are sequential. To overcome this limitation of correlated data and non-stationary distributions, we use an *experience replay* mechanism which randomly samples previous mini-batch of transitions (s_t, a_t, r_t, s_{t+1}) from the dataset \mathcal{D} , and therefore smooths the training distribution over many historical data. It is straightforward to integrate the above Deep RL approach into Eq.1. The binary action vector $a_t \in \mathcal{A}$ is augmented to $\max_{a_{t+1}} Q(s_{t+1}, a_{t+1}, \theta)$ and therefore $\sum_{d=1}^m a_t P_{d,t}^-$ is optimally controlled. Specifically, rather than enforcing the constraints on the time window required by a specific device d and the comfort of end-users considered in A1, Eq.8, our idea is to encapsulate them in the reward function, $r_t(\lambda_t^+, \lambda_t^-, P_{i,d}^-)$, which is further detailed in Section IV.A.

B. Deep Policy Gradient (DPG)

Recently, it has been shown that policy gradient methods are able to decrease the time needed for convergence in continuous games [24], [25]. From an architectural perspective, the neurons from the output layer of the DNN (with parameters θ) corresponding to DPG, instead of estimating the $Q(s_t, a, \theta)$, $\forall a \in \mathcal{A}$ as in DQN, they estimate the probability to take action a in a specific state s_t , such as $p(a|s_t, \theta)$, $\forall a \in \mathcal{A}$. This offers a clear advantage to DPG over DQN when there is a need to perform multiple actions simultaneously, as all actions can be sampled and executed simultaneously in the game using their own probability.

In the policy gradient context, the approximate optimization problem defined in Eq. 1 or Eq. 7 is an equivalent of maximizing the total expected reward of a parameterized model under a policy π , as follows

$$\text{maximize } \mathbb{E}_{x \sim p(x|\theta)}[\mathcal{R}|\pi] \quad (13)$$

In the DPG context, the parameterized model is the DNN. Thus, the DNN becomes a probability density function over its inputs (the game states), i.e. $f(x)$, leading Eq. 13 to the following optimization problem

$$\text{maximize } \mathbb{E}_{x \sim p(x|\theta)}[f(x)] \quad (14)$$

As shown in [26], [27], the unbiased gradient estimation uses $f(x)$ as a score function yielding

$$\begin{aligned} \nabla_{\theta} \mathbb{E}_x[f(x)] &= \nabla_{\theta} \int dx p(x|\theta) f(x) = \int dx \nabla_{\theta} p(x|\theta) f(x) \\ &= \int dx p(x|\theta) \frac{\nabla_{\theta} p(x|\theta)}{p(x|\theta)} f(x) \\ &= \int dx p(x|\theta) \nabla_{\theta} \log p(x|\theta) f(x) \\ &= \mathbb{E}_x[f(x) \nabla_{\theta} \log p(x|\theta)] \end{aligned} \quad (15)$$

where $\frac{\partial}{\partial \theta} = \nabla_{\theta}$ denote the first-order partial derivative over the output data. Intuitively, to solve the gradient of Eq. 15, first we have to take samples of $x_i \sim p(x|\theta)$ and to compute the estimated gradient, such that $\hat{g}_i^{\theta} = f(x_i) \nabla_{\theta} \log p(x_i|\theta)$. Moving in the \hat{g}_i direction increases the log-probability of that particular sample x_i proportional with the reward associated with it, $f(x_i)$. In other words, this practically shows how good is that sample. As in policy gradient the reward is available at the end of a game, these samples are collected in a trajectory, i.e. $\tau = (s_0, a_0, r_0, \dots, s_{T-1}, a_{T-1}, r_{T-1})$. To compute the gradient of a trajectory, we need to calculate and differentiate the density $p(\tau|\theta)$ with respect to θ as follows:

$$p(\tau|\theta) = p(s_0) \prod_{t=0}^{T-1} [\pi(a_t|s_t, \theta) p(s_{t+1}|s_t, a_t)] \quad (16)$$

By taking the log-probability of Eq. (16), we obtain

$$\begin{aligned} \log p(\tau|\theta) &= \log p(s_0) + \sum_{t=0}^{T-1} \left[\log \pi(a_t|s_t, \theta) \right. \\ &\quad \left. + \log p(s_{t+1}|s_t, a_t) \right] \end{aligned} \quad (17)$$

Taking the derivative of Eq. (17) with respect to θ leads to

$$\frac{\partial}{\partial \theta} \log p(\tau|\theta) = \frac{\partial}{\partial \theta} \sum_{t=0}^{T-1} \log \pi(a_t|s_t, \theta) \quad (18)$$

Finally, we can write the gradient update \hat{g}_{τ}^{θ} for parameters θ after considering a trajectory τ as

$$\hat{g}_{\tau}^{\theta} \propto \mathcal{R}_{\tau} \frac{\partial}{\partial \theta} \sum_{t=0}^{T-1} \log \pi(a_t|s_t, \theta) \quad (19)$$

V. IMPLEMENTATION DETAILS

A. Network Architecture

Aiming to have a fair comparison between DQN and DPG, the architecture of the deep neural networks used is similar for both models and it has the following characteristics. Each reinforcement learning state (encoded in the input layer), is given by a time-window of two consecutive time steps. Thus, in the case of the peak reduction problem the input layer has 11 neurons, i.e. time step t , and base load, PV, AC state, EV and dishwasher at $t - 1$ and t . Please note that with the exception of base load and generation which are fixed, the other state components are given by the dynamically adapted values (the ones obtained during the learning process) and not the initial ones measured by the smart meters. For the cost reduction problem, the input layer has an extra neuron which is used to encode the ToU tariff. Furthermore, the networks have three layers of hidden neurons, each layer having 100 neurons with Rectifier Linear Units (ReLU) as activation function.

The output layer differs for DQN and DPG. For DQN the output layer has 8 neurons, each neuron representing the Q-value of a combined action. Each combined action is a possible combination of the actions of the three flexible devices¹, i.e. stop air conditioner (a_1), electric vehicle on/off (a_2), dishwasher on/off (a_3). By contrast, the DPG output layer has just three neurons, each neuron representing a device action. More precisely, it gives the probability to perform the action associated with the flexible device for the specific input state. This is a clear advantage of DPG over DQN as it scales linearly with the number of flexible devices.

Hyper-parameters settings: In all experiments performed, the learning rate is set to $\alpha = 10^{-2}$, the discount factor to $\gamma = 0.99$, and $\eta = 0.01$. We train the models for 5000 episodes, where an episode is composed by 20 randomly chosen days. The weights update is performed after every two episodes. The final policy is kept as the output of the learning process.

B. The reward vectors for DRL

Regarding the multi-objective optimization problems solved in this paper, an accurate reward function is computed at the end of the day, instead of at each time step of the day. Thus, we derived a simple multiple-task joint reward with three reward components:

Component 1: For all $\tau = (s_t, a_t, r_t)$ the reward vectors will be able to control the actions of the three types of flexible consumption, and therefore the total shiftable and scalable load in a household $\sum_{d=1}^m a_t P_{d,t}^-$, using differentiated rewards

$$r_{a_1} = \begin{cases} -n_{a_1^+} & \text{if } n_{a_1^+} > 10 \\ \zeta_1 & \text{if } n_{a_1^+} \in [1, 10]; \\ \zeta_2 & \text{if } n_{a_1^+} < 1 \end{cases}; \quad r_{a_3} = \begin{cases} -n_{a_3^+} & \text{if } n_{a_3^+} > 2 \\ \zeta_1 & \text{if } n_{a_3^+} \in [1, 2] \\ \zeta_2 & \text{if } n_{a_3^+} < 1 \end{cases}$$

$$r_{a_2} = \begin{cases} -4|n_{a_2^t} - n_{a_2^+}| & \text{if } n_{a_2^+} \neq n_{a_2^t}, \forall n_{a_2^t} \in \mathbb{N} \\ n_{a_2^+} & \text{if } n_{a_2^+} = n_{a_2^t}, \forall n_{a_2^t} \in \mathbb{N} \end{cases} \quad (20)$$

¹The number of neurons in the output layer of DQN is exponentially correlated with respect to the number of flexible devices.

where $n_{a_1^+}$, $n_{a_2^+}$, and $n_{a_3^+}$ represent how many times the action corresponding with the flexible device is performed, and $n_{a_2^t}$ is the targeted number of loads per day for the electric car. The choice of ζ_1 and ζ_2 coefficients was based on a trial and error procedure. The obtained values are $\zeta_1 = 40$ and $\zeta_2 = -50$.

Component 2: Controlling the total energy consumption defined in Eq. 7 is done as follows

$$r = \begin{cases} -3\zeta_2 + 4[\max(P^-) - \max(\tilde{P}^-)] & \text{if } \max(\tilde{P}^-) < \max(P^-) \\ -3\zeta_1 - 1 & \text{otherwise} \end{cases} \quad (21)$$

Further on, shifting the consumption through the time when there is more generation Eq. 6.

$$r = \begin{cases} \frac{\zeta_1}{2} - |\min(\tilde{P}^-)| & \text{if } \tilde{P}^- < 0 \\ -\frac{\zeta_2}{2} & \text{otherwise} \end{cases} \quad (22)$$

The control of AC under the A2, Eq. 8 is given by

$$r = \begin{cases} \frac{\zeta_1}{8} + 2[\max(\tilde{P}_{AC}^-) - \max(P_{AC}^-)] & \text{if } \tilde{P}^- < 0 \\ -\frac{\zeta_2}{10} & \text{otherwise} \end{cases} \quad (23)$$

Component 3: Controlling the total cost \mathcal{C} , defined in Eq. 1.

$$r = \begin{cases} 5|\tilde{\mathcal{C}} - \mathcal{C}| & \text{if } \tilde{\mathcal{C}} < \mathcal{C} \\ -3\zeta_1 - 1 & \text{otherwise} \end{cases} \quad (24)$$

The agent must learn multiple tasks consecutively with the goal of optimizing performance across all previously learned tasks. So, we used for solving Eq. 7 the **Component 1** and **2** of the reward, while for Eq. 1 the **Component 1** and **3**.

The joint reward components could be easily generalized to perform an arbitrary number of tasks. However, the range intervals for $n_{a_1^+}$ and $n_{a_3^+}$ considered in Eq. 20 as well as the *positive* and *negative* coefficients (i.e. ζ_1 and ζ_2) used in Eq. 20-24 are dependent on the application. Also in Eq. 20 the range of $n_{a_1^+}$ and $n_{a_3^+}$ may be enlarged if comfort limits are relaxed. Algorithm 1 exemplifies on DPG, how DRL can be implemented. We have implemented both methods, DQN and DPG, in Python.

Algorithm 1 Deep Policy Gradient (DPG) - estimating Eq.13

- 1: initialize model: hyper-parameters (α, γ, ζ)
 - 2: initialize model: DNN with random weights θ
 - 3: initialize game: first state s from a random day
 - 4: **for** *iteration* = 1 to *arbitrary number* **do**
 - 5: sample actions $p(a_1, a_2, a_3 | \theta, s)$ with DNN
 - 6: collect probabilities $p(a_1, a_2, a_3 | \theta, s)$ in A
 - 7: collect hidden neurons values in H from DNN
 - 8: collect s in S
 - 9: execute actions a_1, a_2, a_3 and move to next state s'
 - 10: collect reward r in R from game
 - 11: **if** episode is finished **then**
 - 12: compute discounted rewards R_d from R
 - 13: estimate gradients from $A * R_d, \theta, S$, and H (Eq. 19)
 - 14: update θ with the estimated gradient
 - 15: empty A, R, S , and H
 - 16: **end if**
 - 17: **if** current day ends **then**
 - 18: reset game: first state s' from a random day
 - 19: **end if**
 - 20: set $s = s'$
 - 21: **end for**
-

TABLE I
DAILY PEAK VALUE AT THE BUILDING LEVEL (B) AVERAGED OVER ONE YEAR WITH 15 MINUTES RESOLUTION VERSUS OPTIMIZED PEAK VALUE USING DQN AND DPG METHODS

	Method	B_I		B_{II}		B_{III}	
		Mean (μ)	St.dev. (Σ)	Mean (μ)	St.dev. (Σ)	Mean (μ)	St.dev. (Σ)
Peak [kW]	-	3.81	1.72	3.77	2.32	4.55	1.52
Optimized peak [kW]	DQN	2.72	1.45	2.72	1.21	3.59	1.41
	DPG	2.55	1.36	2.49	1.13	3.12	1.31

VI. RESULTS AND DISCUSSION

In this section, we validate our proposed methods and we analyze their performance on a large real-world database recorded by Pecan Street.Inc. First, the database is described. Then, numerical results are given for both problems, i.e. peak reduction and cost minimization for various number of buildings.

A. Data set characteristics

1) *Buildings pattern*: To validate our proposed method we used the Pecan Street dataset. The disaggregated customer energy data contains up to 90 million unique electricity consumption records per day, which are used in order to build specific device patterns. Figure 2(a), 2(b), and 2(c) show three different building patterns averaged over the year 2015 with 15 minutes resolution. In these patterns the solar generation uncertainty as well as the people behavior characteristics are notable, e.g. even if all three buildings have an electric vehicle, just in the case of the third building, Fig. 2(c), it is used frequently. In our experiments, we have used the data between 27th October 2012 and 3rd September 2016.

2) *Price data*: We use the time-of-use (ToU) tariff provided by the local grid operator Austin Energy for customers who live inside the City of Austin, Texas². The λ^- summer rates are composed by on-peak, mid-peak and off-peak hours and the winter tariff has the mid-peak and off-peak hours components. There is also a difference between weekend and working-days tariff. Additionally, the self-generating customers are receiving an amount that is being paid by the utility for solar generation, called the value of solar tariff (VOST).

B. Numerical results - Peak reduction problem

The numerical results in terms of peak reduction at the single building level are showed in Table I and Figure 2 for three different buildings (B_I , B_{II} and B_{III}), over one year with 15 minutes resolution.

C. Numerical results - Cost minimization problem

The results for the cost minimization problem are summarized in Table II. The difference between the cost minimization solutions obtained for every building correlated with their average electrical patterns (see Figure 2) give as a first indication about the individual capabilities of the end-users to adopt a more conservative behavior.

²<http://austinenergy.com/wps/portal/ae/residential/rates/residential-electric-rates-and-line-items> (Last visit: 30 October 2016)

TABLE II
DAILY COST MINIMIZATION RESULTS AT THE BUILDING LEVEL AVERAGE OVER ONE YEAR WITH 15 MINUTES RESOLUTION USING DQN AND DPG METHODS

	Method	B_I		B_{II}		B_{III}	
		Mean (μ)	St.dev. (Σ)	Mean (μ)	St.dev. (Σ)	Mean (μ)	St.dev. (Σ)
Peak [kW]	-	3.81	1.72	3.77	2.32	4.55	1.52
Peak [kW]	DQN	3.12	1.51	3.48	2.13	3.62	1.34
reduction	DPG	2.97	1.46	2.69	1.14	3.17	1.29
Cost [\$/day]	-	2.31	3.09	1.93	2.23	3.13	3.85
Minimized cost [\$/day]	DQN	2.19	3.01	1.91	2.18	2.85	3.62
	DPG	2.08	2.78	1.79	2.06	2.73	3.38

TABLE III
PEAK REDUCTION – DAILY OPTIMIZATION RESULTS AT DIFFERENT LEVELS OF AGGREGATION AVERAGE OVER ONE YEAR WITH 15 MINUTES RESOLUTION USING DQN AND DPG METHODS

	Method	Number of buildings					
		10		20		48	
		Mean (μ)	St.dev. (Σ)	Mean (μ)	St.dev. (Σ)	Mean (μ)	St.dev. (Σ)
Peak [kW]	-	59.79	6.12	124.72	10.28	281.88	14.32
Optimized peak [kW]	DQN	49.67	5.62	106.84	7.49	238.12	12.98
	DPG	41.74	5.08	93.83	7.29	213.01	12.02

As it can be observed in Table II and in Figure 2, a secondary advantage of solving the cost minimization problem is its impact on solving of the peak reduction problem also. Therein, the best results in terms of both, peak reduction and cost reduction, are obtained for building B_{III} using DPG.

D. Scalability and learning capabilities of DRL

To test whether good estimations occur in practice and at scale, we investigate the performance of our proposed methods, in three cases with different numbers of customers using data from the Pecan Street smart grid test-bed. Specifically, we are investigating and analyzing the corresponding results using DQN and DPG methods for 10, 20 and 48 buildings, respectively. Tables III and IV show that our proposed approach is scalable for both, peak reduction and cost minimization, respectively. More than that, they show that at the aggregated level, when more customers are taking the cost minimization problem into consideration, this solves implicitly also the peak reduction problem. Same as at the building level, DPG is more stable than DQN, and achieves a better performance. Overall, in the case of the cost reduction problem for 48 buildings, DPG reduces the peak with 26.3% and minimizes the cost with 27.4%, while DQN reduces the peak with just 9.6% and minimizes the cost with just 14.1%. To visualize how DPG performs, in Figure 3 we depict the unoptimized and the optimized annualized energy costs for each of the 48 buildings.

To visualize how DPG performs, in Figure 3 we depict the unoptimized and the optimized annualized energy costs for each of the 48 buildings. We can observe that the buildings behave very differently and in some cases DPG is capable to halve the yearly cost, while in other cases it succeeds to reduce the cost with just a few percentage points.

Convergence capabilities of DPG: The convergence is assessed through many iterations over episodes. For example, the

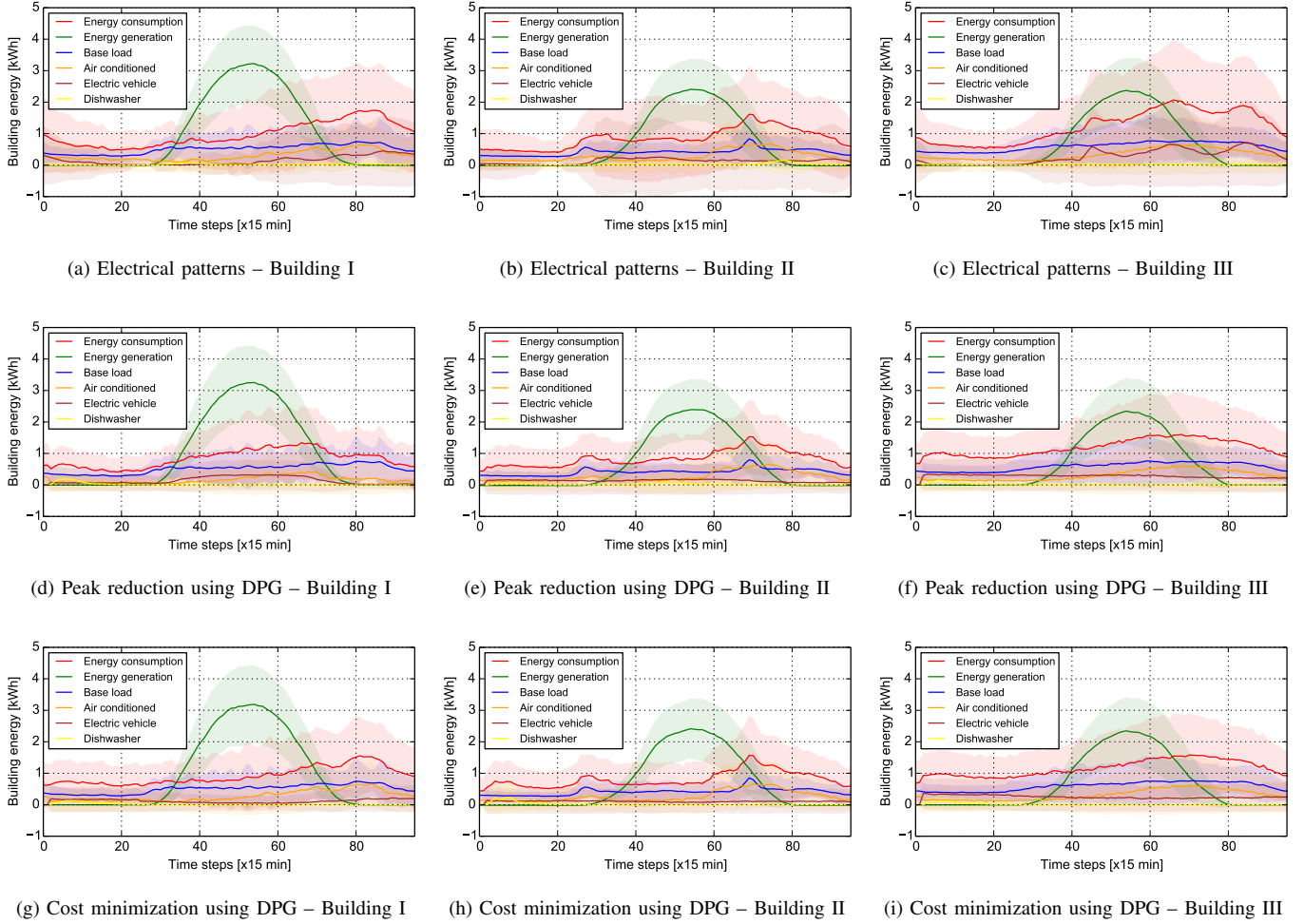


Fig. 2. (a), (b) and (c) represent different building electrical patterns averaged over one year (solid line) with 15 minute resolution, followed by their standard deviation (shadow area). Their yearly average optimization results using Deep Policy Gradient method for the peak reduction problem are depicted in Fig.2 (d), (e) and (f) whether the cost minimization results are showed in Fig.2. (g), (h) and (i).

TABLE IV

COST REDUCTION – DAILY OPTIMIZATION RESULTS AT DIFFERENT LEVELS OF AGGREGATION AVERAGE OVER ONE YEAR WITH 15 MINUTES RESOLUTION USING DQN AND DPG METHODS

	Method	Number of buildings					
		10		20		48	
		Mean (μ)	St.dev. (Σ)	Mean (μ)	St.dev. (Σ)	Mean (μ)	St.dev. (Σ)
Peak [kW]	-	59.79	6.12	124.72	10.28	281.88	14.32
Peak [kW] reduction	DQN	54.85	5.93	116.72	9.24	254.67	13.21
	DPG	44.91	4.80	92.41	7.74	207.73	11.48
Cost [\$/day]	-	57.79	20.90	118.03	30.01	231.27	38.76
Minimized cost [\$/day]	DQN	47.71	17.83	93.68	24.18	198.51	32.67
	DPG	44.35	16.01	82.71	21.48	167.70	28.62

learning capabilities of DPG method in terms of peak reduction and their corresponding reward function for a building are showed in Figure 4. Each episode represents an average value over 20 randomly chosen days. Initially, we may observe that the reward increases fast, while after about 1000 episodes the reward, as expected, increases much slower. Therefore, after approximately 1000 episodes the average peak value and the optimized average peak value using the Deep Policy Gradient method converge. Still, the long-term reward expectation, as

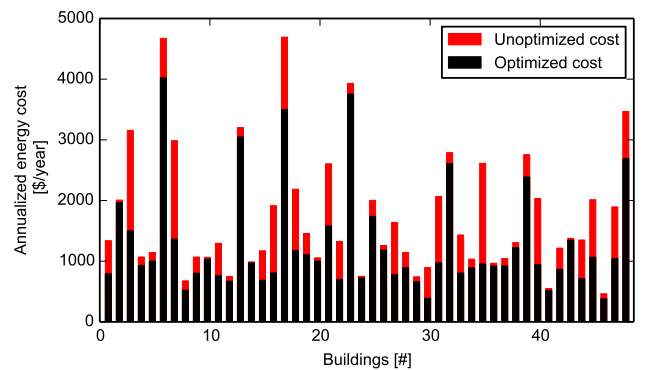


Fig. 3. Yearly savings per buildings when cost optimization is performed at the aggregated level on 48 buildings using Deep Policy Gradient (DPG) method.

was expressed in Eq. (13), is increasing until approximately 2500 episodes.

Computational time requirements: Both DRL variants have the advantage of handling naturally much larger continuous state spaces, leading to better performance. In comparison with heuristic methods (e.g. PSO), after DRL learns how to act, it can make decisions (e.g. choosing the optimal control action)

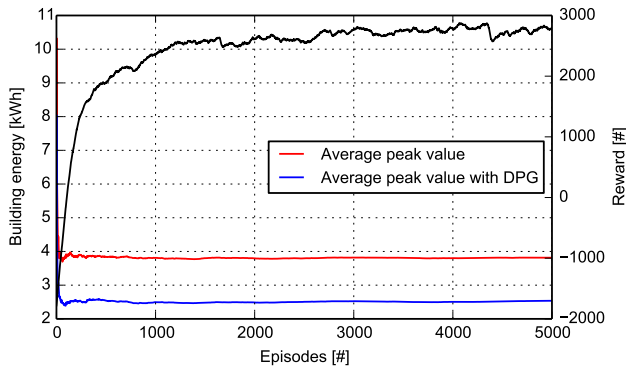


Fig. 4. Learning capabilities of Deep Policy Gradient method in terms of peak reduction and their corresponding reward function for a building (i.e. Fig.2 Building I). Every epoch represent an average value over 20 random days.

in a few milliseconds, while PSO needs to re-run the costly optimization process for every decision.

VII. CONCLUSIONS

In this paper, we proposed the use of Deep Reinforcement Learning, as a hybrid method which combines Reinforcement Learning with Deep Learning, with the aim of conceiving an on-line optimization for the scheduling of electricity consuming devices in residential buildings and aggregations of buildings. We have shown that a single agent, empowered with a suitable learning algorithm, can solve many challenging tasks. We proposed two optimization methods, Deep Q-learning and Deep Policy Gradient, to solve the same sequential decision problems at both the building level and the aggregated level. At both levels, we showed that Deep Policy Gradient is more suited to perform on-line scheduling of energy resources than Deep Q-learning. We explored and validated our proposed methods using the large Pecan Street database. Both methods are able to successfully perform either the minimization of the energy cost or the flattening of the net energy profile. For the minimization of the energy cost, a variable electricity price signal is investigated to incentivize customers to shift their consumption to low-price, off-peak periods.

ACKNOWLEDGMENT

This research has been partly funded by the NL Enterprise Agency under the TKI SG-BEMS project of Dutch Top Sector and by the European Union's Horizon 2020 project INTER-IoT (grant number 687283).

REFERENCES

- [1] M. R. Alam, M. St-Hilaire, and T. Kunz, "Computational methods for residential energy cost optimization in smart grids: A survey," *ACM Comput. Surv.*, vol. 49, no. 1, pp. 2:1–2:34, Apr. 2016.
- [2] A. Barbato and A. Capone, "Optimization models and methods for demand-side management of residential users: A survey," *Energies*, vol. 7, no. 9, pp. 5787 – 5824, 2014.
- [3] E. Loukarakis, C. J. Dent, and J. W. Bialek, "Decentralized multi-period economic dispatch for real-time flexible demand management," *IEEE Transactions on Power Systems*, vol. 31, no. 1, pp. 672–684, 2016.
- [4] A. H. Mohsenian-Rad and A. Leon-Garcia, "Optimal residential load control with price prediction in real-time electricity pricing environments," *IEEE Transactions on Smart Grid*, vol. 1, no. 2, pp. 120–133, 2010.

- [5] N. G. Paterakis, O. Erdinc, I. N. Pappi, A. G. Bakirtzis, and J. P. S. Catalão, "Coordinated operation of a neighborhood of smart households comprising electric vehicles, energy storage and distributed generation," *IEEE Transactions on Smart Grid*, vol. 7, no. 6, pp. 2736–2747, Nov 2016.
- [6] J. Vardakas, N.Zorba, and C. Verikoukis, "A survey on demand response programs in smart grids: Pricing methods and optimization algorithms," *IEEE Communication Surveys Tutorials*, vol. 17, no. 1, pp. 152–178, 2015.
- [7] L. A. Hurtado, E. Mocanu, P. H. Nguyen, M. Gibescu, and W. L. Kling, "Comfort-constrained demand flexibility management for building aggregations using a decentralized approach," in *International Conference on Smart Cities and Green ICT Systems (SMARTGREENS)*, May 2015, pp. 1–10.
- [8] C. J. C. H. Watkins and P. Dayan, "Technical note: Q-learning," *Journal of Machine Learning Research*, vol. 8, no. 3-4, pp. 279–292, May 1992.
- [9] R. S. Sutton and A. G. Barto, *Introduction to Reinforcement Learning*, 1st ed. Cambridge, MA, USA: MIT Press, 1998.
- [10] D. Ernst, M. Glavic, F. Capitanescu, and L. Wehenkel, "Reinforcement learning versus model predictive control: A comparison on a power system problem," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 39, no. 2, pp. 517–529, April 2009.
- [11] D. O'Neill, M. Levorato, A. Goldsmith, and U. Mitra, "Residential demand response using reinforcement learning," in *First IEEE International Conference on Smart Grid Communications (SmartGridComm)*, 2010, pp. 409–414.
- [12] B. G. Kim, Y. Zhang, M. van der Schaar, and J. W. Lee, "Dynamic pricing and energy consumption scheduling with reinforcement learning," *IEEE Transactions on Smart Grid*, vol. 7, no. 5, pp. 2187–2198, 2016.
- [13] F. Ruelens, B. J. Claessens, S. Vandael, S. Iacovella, P. Vingerhoets, and R. Belmans, "Demand response of a heterogeneous cluster of electric water heaters using batch reinforcement learning," in *Power Systems Computation Conference (PSCC)*, 2014, pp. 1–7.
- [14] H. Berlink and A. H. R. Costa, "Batch reinforcement learning for smart home energy management," in *Proceedings of the 24th International Conference on Artificial Intelligence*, ser. IJCAI'15, 2015.
- [15] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015.
- [16] V. François-Lavet, D. Taralla, D. Ernst, and R. Fonteneau, "Deep reinforcement learning solutions for energy microgrids management," in *European Workshop on Reinforcement Learning (EWRL 2016)*, 2016.
- [17] E. Mocanu, P. H. Nguyen, W. L. Kling, and M. Gibescu, "Unsupervised energy prediction in a smart grid context using reinforcement cross-building transfer learning," *Energy and Buildings*, vol. 116, pp. 646 – 655, 2016.
- [18] J. L. Bode, M. J. Sullivan, and J. H. Eto, "Measuring short-term air conditioner demand reductions for operations and settlement." *Berkeley: LBNL*, 2012.
- [19] M. Kraning, E. Chu, J. Lavaei, and S. Boyd, "Dynamic network energy management via proximal message passing," *Foundations and Trends in Optimization*, vol. 1, no. 2, pp. 73–126, 2014.
- [20] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics (AISTATS-11)*, vol. 15, 2011, pp. 315–323.
- [21] A. Maas, A. Hannun, and A. Ng, "Rectifier nonlinearities improve neural network acoustic models." 2013.
- [22] M. D. Zeiler, M. Ranzato, R. Monga, M. Mao, K. Yang, Q. V. Le, P. Nguyen, A. Senior, V. Vanhoucke, J. Dean, and G. E. Hinton, "On rectified linear units for speech processing," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, May 2013, pp. 3517–3521.
- [23] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 1026–1034.
- [24] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," *CoRR*, vol. abs/1602.01783, 2016.
- [25] J. Schulman, P. Moritz, S. Levine, M. I. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," *CoRR*, vol. abs/1506.02438, 2015.

- [26] J. Peters and S. Schaal, "Reinforcement learning of motor skills with policy gradients," *Neural Networks*, vol. 21, no. 4, pp. 682 – 697, 2008.
- [27] H. Bou-Ammar, E. Eaton, P. Ruvolo, and M. E. Taylor, "Online multi-task learning for policy gradient methods," in *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, 2014, pp. 1206–1214.