Deep Reinforcement Learning for Direct Load Control in Distribution Networks

Shahab Bahrami, Yu Christine Chen, and Vincent W.S. Wong
Department of Electrical and Computer Engineering, The University of British Columbia, Vancouver, Canada
email: {bahramis, chen, vincentw}@ece.ubc.ca

Abstract—**Direct load control enables load aggregators in distribution networks to remotely curtail customers' appliances during peak time periods. This paper proposes a direct load control algorithm for residential customers, while accounting for the uncertainties in the customers' discomfort from curtailing their demand as well as the operational constraints imposed by the distribution network. We model the load control problem as a Markov decision process (MDP). Solving such an MDP is challenging due to the ac power flow equations and the unknown dynamics of the system states (i.e., price, demand, and customer's discomfort). We develop a deep reinforcement learning algorithm based on the actor-critic method that enables the load aggregator to consider the distribution network constraints and the consequences of its past decisions to update the neural network parameters for the policy and value function without any knowledge of the system dynamics. Simulations are performed on an IEEE 85-bus test feeder with 59 households. Results show that the load aggregator learns to reduce the peak load by 16.7%, while taking into account the distribution network constraints. Also, the customers' cost is decreased by 26.6% on average; thereby reaching a win-win outcome.**

I. Introduction

Proper management of distribution networks is crucial to meet the electricity demand during peak load periods. A well-designed direct load control program is a viable approach for load aggregators to target peak reduction by remotely adjusting the operation of appliances in residential households [1]. It can prevent large-scale emergency outages, high network infrastructure investment for additional generation and transmission, and voltage drop during peak load periods. At the same time, customers can benefit from lower electricity bill payments.

Implementing a direct load control program involves challenges for the load aggregator due to the uncertainty in the electricity prices and customer load demand. Also, the load aggregator usually does not have information about the customers' discomfort from curtailing their load demand. Moreover, the load aggregator should account for the operational constraints imposed by the distribution network, since the power flow is sensitive to the load changes.

There are several studies in the literature that address the uncertainties in the price, load demand, and customers' preferences using mechanisms such as stochastic optimization [2], [3], robust optimization [4], and dynamic programming [5]. These techniques, however, require knowledge of the uncertain parameters' stochastic process, which may not be available in practice. To address this challenge, there have been some efforts on applying deep reinforcement learning for load control in microgrids [6], residential buildings [7]–[9], and electric vehicles charging stations [10], [11]. Deep reinforcement learning does not require knowledge of the parameters' stochastic process. However, a load control policy may not yield a feasible power flow solution in the distribution network. Existing studies do not consider the constraints imposed by the topology and operation of the distribution network.

In this paper, we apply deep reinforcement learning to develop a load control algorithm for the load aggregator in a distribution network. We take into account the uncertainty in price, load demand variation, and customers' discomfort from curtailing their desirable demand. The main challenge that we address is to include the distribution network constraints in the learning process to guarantee that the obtained load control action corresponds to a feasible power flow in the distribution network. The contributions of this paper are as follows:

- *Load Control Algorithm Design:* We apply an actor-critic-based [12] deep reinforcement learning [13], which is more robust than actor-only methods (such as the policy evaluation [6], [7]) and faster than critic-only methods (such as the Q-learning [8]–[11]). It enables the load aggregator to gradually update the neural network parameters associated with the policy and value function based on its experience from the past load control decisions.
- *Distribution Network Constraints:* A challenge in learning algorithm design is to update the load control policy considering the ac power flow equations. To address the non-convexity of the power flow equations, we apply convex relaxation and transform the problem of updating the neural network parameters into a sequence of semidefinite programs (SDPs). The load aggregator solves an SDP to choose the *global optimal* load control action under the given policy and distribution network constraints.
- *Reducing Peak Load and Customers' Cost:* We evaluate the performance of our proposed algorithm in reducing the peak load and the average cost of 59 households in an IEEE 85-bus test feeder. Compared with the benchmark of not performing load control, our results show that the load aggregator reduces the peak load by 16.7% and the households' average cost is decreased by 26.6%.

II. System Model

Consider a distribution feeder consisting of N buses. Let $\mathcal{N} = \{1, \ldots, N\}$ denote the set of buses. We assume that bus N corresponds to the substation bus and bus $n \in \mathcal{N}^-$ corresponds to household n, where $\mathcal{N}^- = \{1, \ldots, N-1\}$. Let $\mathcal{L} \subseteq \mathcal{N} \times \mathcal{N}$ denote the set of transmission lines. Each household is equipped with an energy consumption controller (ECC), which is responsible for scheduling the appliances in that household. The ECCs are connected to a load aggregator via a two-way communication network, which enables the load aggregator to remotely control the operation of some appliances in the households based on an agreement. We consider long-term load control (e.g., several weeks) and approximate the load control problem with an infinite operation horizon. We consider a discrete set $\mathcal{T} = \{1, 2, \dots\}$ of time slots, each with equal duration, e.g., 15 minutes per time slot.

A. Household Action and State Models

The active power demand $P_n(t)$ for household $n \in \mathcal{N}^-$ in time slot $t \in \mathcal{T}$ consists of the controllable load demand $P_n^{\rm c}(t)$ and the base load demand $P_n^{\rm b}(t)$, which is uncontrollable. We have $P_n(t) = P_n^{\rm b}(t) + P_n^{\rm c}(t)$. The household's appliances may include electromagnetic devices require reactive power. We consider the overall power factor $\xi_n \in [-1, 1] \backslash \{0\}$ for household n, which is known *a priori* by the ECC. The reactive power demand of household n in time slot t is obtained as $Q_n(t) = P_n(t)\,\text{sign}\{\xi_n\}\sqrt{\frac{1}{\xi_n^2} - 1}$, where $\text{sign}\{\xi_n\} = 1$ with lagging power factor and $\text{sign}\{\xi_n\} = -1$ otherwise.

The *action* for household $n \in \mathcal{N}^-$ in time slot t is defined as the scheduled controllable load $P_n^{\rm c}(t)$. Let $P_n^{\rm c,min}(t)$ denote the lower bound for the controllable load of household n in time slot t. Household n incurs a discomfort cost $d_n(P_n^{\rm c}(t),\, P_n^{\rm c,des}(t))$ in time slot t, which expresses the consumer's dissatisfaction with changing its controllable load demand from the desirable value $P_n^{\rm c,des}(t)$ to the scheduled value $P_n^{\rm c}(t)$. We take into account the uncertainty in the discomfort cost of household n by assuming that the load aggregator becomes aware of the discomfort cost $d_n(P_n^{\rm c}(t),\, P_n^{\rm c,des}(t))$ at the end of time slot t, *after* scheduling the controllable loads. We assume that the base load $P_n^{\rm b}(t)$, and parameters $P_n^{\rm c,min}(t)$ and $P_n^{\rm c,des}(t)$ for household n are revealed to the load aggregator at the beginning of time slot t, *before* scheduling the controllable loads. We define the *state* of household n in time slot t as vector $\boldsymbol{s}_n(t) = (P_n^{\rm b}(t),\, P_n^{\rm c,min}(t),\, P_n^{\rm c,des}(t))$.

Let $\rho(t)$ denote the electricity price in time slot $t \in \mathcal{T}$. We consider a *stationary* decision making for the load aggregator, which only depends on the system state $\boldsymbol{s}(t) = (\boldsymbol{s}_n(t),\, n \in \mathcal{N}^-,\, \rho(t))$, including the state of households and the electricity price. Considering stationary load scheduling, we can replace time index t by the state index $\boldsymbol{s}$ in the variables and parameters. We model the load control problem as a Markov decision process (MDP), where the system state in the next time slot $t+1$ can be inferred from the state and action in the current time slot t [14]. The load aggregator observes the system state $\boldsymbol{s}$ and broadcasts the scheduled demand to the households' ECC. The load aggregator curtails the energy use in the households. Thus, in state $\boldsymbol{s}$, we have

$$P_n^{\rm c,min}(\boldsymbol{s}) \le P_n^{\rm c}(\boldsymbol{s}) \le P_n^{\rm c,des}(\boldsymbol{s}), \;\; n \in \mathcal{N}^-. \tag{1}$$

B. Power Flow Feasible Space

Let Y denote the distribution feeder admittance matrix. For bus $n \in \mathcal{N}$, let e_n denote the n^{th} basis column vector in $\mathbb{R}^N$ and $Y_n = e_n e_n^{\rm T} Y$. We use the lumped-element Π model for transmission lines. Let y_{nm} and $\overline{y}_{nm}$ denote the series and shunt admittance values at bus n for the line $(n,m) \in \mathcal{L}$, respectively. We define $Y_{nm} = (\overline{y}_{nm} + y_{nm}) e_n e_n^{\rm T} - y_{nm} e_n e_m^{\rm T}$, so that the entries (n,n) and (n,m) of Y_{nm} are $\overline{y}_{nm} + y_{nm}$ and $-y_{nm}$, respectively. Other entries of Y_{nm} are zero. We define matrices $\mathbf{Y}_n$, $\overline{\mathbf{Y}}_n$, $\mathbf{Y}_{nm}$, $\overline{\mathbf{Y}}_{nm}$, and $\mathbf{M}_n$ as follows:

$$\mathbf{Y}_n = \frac{1}{2}\begin{bmatrix} \text{Re}\{Y_n + Y_n^{\rm T}\} & \text{Im}\{Y_n^{\rm T} - Y_n\} \\ \text{Im}\{Y_n - Y_n^{\rm T}\} & \text{Re}\{Y_n + Y_n^{\rm T}\} \end{bmatrix}, \tag{2a}$$

$$\overline{\mathbf{Y}}_n = \frac{-1}{2}\begin{bmatrix} \text{Im}\{Y_n + Y_n^{\rm T}\} & \text{Re}\{Y_n - Y_n^{\rm T}\} \\ \text{Re}\{Y_n^{\rm T} - Y_n\} & \text{Im}\{Y_n + Y_n^{\rm T}\} \end{bmatrix}, \tag{2b}$$

$$\mathbf{Y}_{nm} = \frac{1}{2}\begin{bmatrix} \text{Re}\{Y_{nm} + Y_{nm}^{\rm T}\} & \text{Im}\{Y_{nm}^{\rm T} - Y_{nm}\} \\ \text{Im}\{Y_{nm} - Y_{nm}^{\rm T}\} & \text{Re}\{Y_{nm} + Y_{nm}^{\rm T}\} \end{bmatrix}, \tag{2c}$$

$$\overline{\mathbf{Y}}_{nm} = \frac{-1}{2}\begin{bmatrix} \text{Im}\{Y_{nm} + Y_{nm}^{\rm T}\} & \text{Re}\{Y_{nm} - Y_{nm}^{\rm T}\} \\ \text{Re}\{Y_{nm}^{\rm T} - Y_{nm}\} & \text{Im}\{Y_{nm} + Y_{nm}^{\rm T}\} \end{bmatrix}, \tag{2d}$$

$$\mathbf{M}_n = \begin{bmatrix} e_n e_n^{\rm T} & 0 \\ 0 & e_n e_n^{\rm T} \end{bmatrix}. \tag{2e}$$

Let $V_n(\boldsymbol{s})$ denote the voltage phasor of bus n in state $\boldsymbol{s}$. Let $\mathbf{v}(\boldsymbol{s}) = (V_n(\boldsymbol{s}),\, n \in \mathcal{N})$ denote the vector of bus voltages. We separate the real and imaginary parts of $\mathbf{v}(\boldsymbol{s})$ to define variable vector $\mathbf{x}(\boldsymbol{s}) = [\,(\text{Re}\{\mathbf{v}(\boldsymbol{s})\})^{\rm T} \; (\text{Im}\{\mathbf{v}(\boldsymbol{s})\})^{\rm T}\,]^{\rm T}$ in state $\boldsymbol{s}$. We also define variable matrix $\mathbf{W}(\boldsymbol{s}) = \mathbf{x}(\boldsymbol{s})(\mathbf{x}(\boldsymbol{s}))^{\rm T}$ in state $\boldsymbol{s}$. We use matrices in (2a)$-$(2e) to obtain the following constraints imposed by the distribution network [15] in state $\boldsymbol{s}$:

$$P_n(\boldsymbol{s}) = -\text{Tr}\{\mathbf{Y}_n \mathbf{W}(\boldsymbol{s})\}, \qquad n \in \mathcal{N}^- \tag{3a}$$

$$P_n(\boldsymbol{s})\,\text{sign}\{\xi_n\}\sqrt{\frac{1}{\xi_n^2} - 1} = -\text{Tr}\{\overline{\mathbf{Y}}_n \mathbf{W}(\boldsymbol{s})\}, \qquad n \in \mathcal{N}^- \tag{3b}$$

$$0 \le \text{Tr}\{\mathbf{Y}_N \mathbf{W}(\boldsymbol{s})\} \le P_N^{\max}, \tag{3c}$$

$$Q_N^{\min} \le \text{Tr}\{\overline{\mathbf{Y}}_N \mathbf{W}(\boldsymbol{s})\} \le Q_N^{\max}. \tag{3d}$$

$$(V_n^{\min})^2 \le \text{Tr}\{\mathbf{M}_n \mathbf{W}(\boldsymbol{s})\} \le (V_n^{\max})^2, \qquad n \in \mathcal{N} \tag{3e}$$

$$\begin{bmatrix} (S_{nm}^{\max})^2 & \text{Tr}\{\mathbf{Y}_{nm}\mathbf{W}(\boldsymbol{s})\} & \text{Tr}\{\overline{\mathbf{Y}}_{nm}\mathbf{W}(\boldsymbol{s})\} \\ \text{Tr}\{\mathbf{Y}_{nm}\mathbf{W}(\boldsymbol{s})\} & 1 & 0 \\ \text{Tr}\{\overline{\mathbf{Y}}_{nm}\mathbf{W}(\boldsymbol{s})\} & 0 & 1 \end{bmatrix} \succeq 0, \qquad (n,m) \in \mathcal{L} \tag{3f}$$

$$\text{rank}\{\mathbf{W}(\boldsymbol{s})\} = 1. \tag{3g}$$

Constraints (3a) and (3b) represent power balance at buses $n \in \mathcal{N}^-$. Constraints (3c) and (3d) represent the limits on the injected active power $P_N(\boldsymbol{s})$ and reactive power $Q_N(\boldsymbol{s})$ into the substation, respectively. Constraint (3e) shows the limits on the voltage magnitude of bus $n \in \mathcal{N}$. Constraint (3f) is the matrix form of inequality $|S_{nm}(\boldsymbol{s})| \le S_{nm}^{\max}$ for the limit on the apparent power flow in line $(n,m) \in \mathcal{L}$. Constraint (3g) ensures that $\mathbf{W}(\boldsymbol{s})$ is a rank-one matrix.

III. Problem Formulation

In this section, we formulate the load control problem as an MDP with an infinite operation horizon. The system state is the vector $\boldsymbol{s}$. We consider a finite set of states denoted by $\mathcal{S}$. The action in state $\boldsymbol{s} \in \mathcal{S}$ is defined as a tuple $\boldsymbol{\varphi}(\boldsymbol{s}) = (\mathbf{W}(\boldsymbol{s}),\, P_n^{\rm c}(\boldsymbol{s}),\, n \in \mathcal{N}^-)$. The feasible action space $\Phi(\boldsymbol{s})$ is defined by constraints (1) and (3a)$-$(3g). We assume that the load aggregator considers the social cost (i.e., sum of the bill payment and discomfort cost of the households) as the immediate cost in each state $\boldsymbol{s} \in \mathcal{S}$. That is, we have

$$c(\boldsymbol{s},\, \boldsymbol{\varphi}(\boldsymbol{s})) = \textstyle\sum_{n \in \mathcal{N}^-} \Big(\rho(\boldsymbol{s}) P_n(\boldsymbol{s}) + d_n\big(P_n^{\rm c}(\boldsymbol{s}),\, P_n^{\rm c,des}(\boldsymbol{s})\big)\Big). \tag{4}$$

The load aggregator considers a policy in state $\boldsymbol{s} \in \mathcal{S}$ as a probability distribution $\boldsymbol{\pi}(\boldsymbol{s}) = (\pi(\boldsymbol{s},\, \boldsymbol{\varphi}(\boldsymbol{s})),\, \boldsymbol{\varphi}(\boldsymbol{s}) \in \Phi(\boldsymbol{s}))$ that specifies the probability $\pi(\boldsymbol{s},\, \boldsymbol{\varphi}(\boldsymbol{s}))$ of choosing an action $\boldsymbol{\varphi}(\boldsymbol{s})$ in state $\boldsymbol{s}$. A stationary policy is defined as $\boldsymbol{\pi} = (\boldsymbol{\pi}(\boldsymbol{s}),\, \boldsymbol{s} \in \mathcal{S})$. Under a given policy $\boldsymbol{\pi}$, the value function $V^{\boldsymbol{\pi}} : \mathcal{S} \to \mathbb{R}$ returns the *discounted cost* starting with state $\boldsymbol{s}$:

$$V^{\boldsymbol{\pi}}(\boldsymbol{s}) = \mathbb{E}\{Q^{\boldsymbol{\pi}}(\boldsymbol{s},\, \boldsymbol{\varphi}(\boldsymbol{s}))\}, \tag{5}$$

where $\mathbb{E}\{\cdot\}$ is the expectation over selecting different actions $\boldsymbol{\varphi}(\boldsymbol{s}) \in \Phi(\boldsymbol{s})$ under the given policy $\boldsymbol{\pi}$. Function $Q^{\boldsymbol{\pi}}(\boldsymbol{s},\, \boldsymbol{\varphi}(\boldsymbol{s}))$ is the Q-function for action $\boldsymbol{\varphi}(\boldsymbol{s})$ in state $\boldsymbol{s}$ under the given policy $\boldsymbol{\pi}$. For a discount factor $\beta \in [0,\, 1)$, we have

$$\begin{aligned} Q^{\boldsymbol{\pi}}(\boldsymbol{s},\, \boldsymbol{\varphi}(\boldsymbol{s})) &= c(\boldsymbol{s},\, \boldsymbol{\varphi}(\boldsymbol{s})) \\ &\quad + \beta \textstyle\sum_{\boldsymbol{s}' \in \mathcal{S}} \Pr(\boldsymbol{s}' \mid \boldsymbol{s},\, \boldsymbol{\varphi}(\boldsymbol{s})) V^{\boldsymbol{\pi}}(\boldsymbol{s}'), \end{aligned} \tag{6}$$

where $\Pr(\boldsymbol{s}' \mid \boldsymbol{s},\, \boldsymbol{\varphi}(\boldsymbol{s}))$ is the transition probability from system state $\boldsymbol{s}$ to $\boldsymbol{s}'$ with action $\boldsymbol{\varphi}(\boldsymbol{s})$. The load aggregator aims to determine a policy $\boldsymbol{\pi}$ such that the value function in (5) is minimized over all initial states $\boldsymbol{s} \in \mathcal{S}$. It is equivalent to solving the following Bellman optimality equations:

$$\boldsymbol{\mathcal{P}}_1 : \quad V^{\boldsymbol{\pi}}(\boldsymbol{s}) = \underset{\boldsymbol{\varphi}(\boldsymbol{s}) \in \Phi(\boldsymbol{s})}{\text{minimize}} \; \mathbb{E}\{Q^{\boldsymbol{\pi}}(\boldsymbol{s},\, \boldsymbol{\varphi}(\boldsymbol{s}))\}, \;\; \forall\, \boldsymbol{s} \in \mathcal{S}.$$

Solving problem $\boldsymbol{\mathcal{P}}_1$ is challenging, since the transition probabilities between the states may not be available. We develop a model-free learning algorithm that enables the load aggregator to gradually update the policy and value function without any knowledge on the system dynamics. It is difficult to obtain a feasible action due to the rank-one constraint (3g). Hence, we consider a *modified policy* in state $\boldsymbol{s} \in \mathcal{S}$ as a probability distribution $\widetilde{\boldsymbol{\pi}}(\boldsymbol{s}) = (\widetilde{\pi}(\boldsymbol{s},\, \boldsymbol{P}^{\rm c}(\boldsymbol{s})),\, \boldsymbol{P}^{\rm c}(\boldsymbol{s}) \in \Phi^{\rm c}(\boldsymbol{s}))$ that includes the probability $\widetilde{\pi}(\boldsymbol{s},\, \boldsymbol{P}^{\rm c}(\boldsymbol{s}))$ of choosing vector $\boldsymbol{P}^{\rm c}(\boldsymbol{s})$ for scheduling the households' controllable load in feasible space $\Phi^{\rm c}(\boldsymbol{s})$ defined by constraint (1). The load aggregator uses its modified policy to select an action $\boldsymbol{P}^{\rm c}(\boldsymbol{s}) \in \Phi^{\rm c}(\boldsymbol{s})$ in state $\boldsymbol{s}$. With $\boldsymbol{P}^{\rm c}(\boldsymbol{s})$, there may not exist matrix $\mathbf{W}(\boldsymbol{s})$ that satisfies constraints (3a)$-$(3g). The load aggregator perturbs the selected action $\boldsymbol{P}^{\rm c}(\boldsymbol{s})$ to obtain a new action $\widehat{\boldsymbol{P}}^{\rm c}(\boldsymbol{s})$, for which there exists rank-one matrix $\mathbf{W}(\boldsymbol{s})$. A viable approach to obtain $\widehat{\boldsymbol{P}}^{\rm c}(\boldsymbol{s})$ is to project vector $\boldsymbol{P}^{\rm c}(\boldsymbol{s})$ onto the feasible action space. The load aggregator solves the following optimization problem to obtain vector $\widehat{\boldsymbol{P}}^{\rm c}(\boldsymbol{s})$ and matrix $\mathbf{W}(\boldsymbol{s})$:

$$\begin{aligned} \boldsymbol{\mathcal{P}}_2 : \quad & \underset{\widehat{\boldsymbol{P}}^{\rm c}(\boldsymbol{s}),\, \mathbf{W}(\boldsymbol{s})}{\text{minimize}} \quad ||\widehat{\boldsymbol{P}}^{\rm c}(\boldsymbol{s}) - \boldsymbol{P}^{\rm c}(\boldsymbol{s})||_2^2 \\ & \text{subject to constraints (3a)}-\text{(3g)}, \\ & \qquad \widehat{\boldsymbol{P}}^{\rm c}(\boldsymbol{s}) \in \Phi^{\rm c}(\boldsymbol{s}). \end{aligned}$$

Problem $\boldsymbol{\mathcal{P}}_2$ is a nonconvex optimization problem due to the rank-one constraint (3g). We relax constraint (3g) and replace it with constraint $\mathbf{W}(\boldsymbol{s}) \succeq 0$ that enforces matrix $\mathbf{W}(\boldsymbol{s})$ to be positive semidefinite. Furthermore, we define an auxiliary variable $\alpha(\boldsymbol{s})$, such that $||\widehat{\boldsymbol{P}}^{\rm c}(\boldsymbol{s}) - \boldsymbol{P}^{\rm c}(\boldsymbol{s})||_2^2 \leq \alpha(\boldsymbol{s})$, which can be expressed as the following linear matrix inequality:

$$\begin{bmatrix} \alpha(\boldsymbol{s}) & \big(\widehat{\boldsymbol{P}}^{\rm c}(\boldsymbol{s}) - \boldsymbol{P}^{\rm c}(\boldsymbol{s})\big)^{\rm T} \\ \widehat{\boldsymbol{P}}^{\rm c}(\boldsymbol{s}) - \boldsymbol{P}^{\rm c}(\boldsymbol{s}) & \mathbf{I}(\boldsymbol{s}) \end{bmatrix} \succeq 0, \tag{7}$$

where $\mathbf{I}(\boldsymbol{s})$ is an $|\mathcal{N}^-| \times |\mathcal{N}^-|$ identity matrix. We replace the objective function with $\alpha(\boldsymbol{s})$ to transform problem $\boldsymbol{\mathcal{P}}_2$ into the following optimization problem:

$$\begin{aligned} \boldsymbol{\mathcal{P}}_3 : \quad & \underset{\alpha(\boldsymbol{s}),\, \widehat{\boldsymbol{P}}^{\rm c}(\boldsymbol{s}),\, \mathbf{W}(\boldsymbol{s})}{\text{minimize}} \quad \alpha(\boldsymbol{s}) \\ & \text{subject to constraints (3a)}-\text{(3f) and (7)}, \\ & \qquad \mathbf{W}(\boldsymbol{s}) \succeq 0, \\ & \qquad \widehat{\boldsymbol{P}}^{\rm c}(\boldsymbol{s}) \in \Phi^{\rm c}(\boldsymbol{s}). \end{aligned}$$

Problem $\boldsymbol{\mathcal{P}}_3$ is an SDP and can be solved efficiently to obtain action $\boldsymbol{\varphi}(\boldsymbol{s}) = (\widehat{\boldsymbol{P}}^{\rm c}(\boldsymbol{s}),\, \mathbf{W}(\boldsymbol{s}))$, which is feasible for the network. Next, we show that the optimal solution to $\boldsymbol{\mathcal{P}}_3$ is the global optimal solution to $\boldsymbol{\mathcal{P}}_2$.

Theorem 1: *The relaxation gap between problems $\boldsymbol{\mathcal{P}}_2$ and $\boldsymbol{\mathcal{P}}_3$ is zero. That is, the solution matrix $\mathbf{W}(\boldsymbol{s})$ to $\boldsymbol{\mathcal{P}}_3$ is rank-one.*

Proof sketch: We obtain problem $\boldsymbol{\mathcal{P}}_2^{\rm r}$ by relaxing rank-one constraint (3g) in problem $\boldsymbol{\mathcal{P}}_2$. The objective function $||\widehat{\boldsymbol{P}}^{\rm c}(\boldsymbol{s}) - \boldsymbol{P}^{\rm c}(\boldsymbol{s})||_2^2$ can be expressed as a sum of quadratic functions of $P_n^{\rm c}(\boldsymbol{s}),\, n \in \mathcal{N}^-$. Hence, we can interpret problem $\boldsymbol{\mathcal{P}}_2^{\rm r}$ as an optimal power flow (OPF) problem in the underlying distribution network, where the load in bus n has a quadratic cost function. Practical distribution networks (including IEEE test feeders) satisfy the sufficient conditions given in [15, Sec. IV-C] for the network topology and constraints. Thus, the SDP relaxation gap between problems $\boldsymbol{\mathcal{P}}_2$ and $\boldsymbol{\mathcal{P}}_2^{\rm r}$ is zero. Problem $\boldsymbol{\mathcal{P}}_2^{\rm r}$ is equivalent to $\boldsymbol{\mathcal{P}}_3$. This completes the proof sketch. ■

Another challenge to develop a learning algorithm is to deal with a high-dimensional and large state space $\mathcal{S}$ and continuous action space. We consider *parametrized* policy and value function [13]. In particular, we use a deep neural network with parameters vector $\boldsymbol{\vartheta}$, an input layer $\boldsymbol{s}$, and an output layer $V^{\boldsymbol{\pi}}(\boldsymbol{s}, \boldsymbol{\vartheta})$ to obtain the optimal value function. Also, we use a deep neural network with parameters vector $\boldsymbol{\theta}$, an input layer $\boldsymbol{s}$, and an output layer with softmax function to obtain a *discrete* probability distribution as the modified policy. Instead of updating the value function and policy directly, the load aggregator determines the optimal values of the neural network parameters $\boldsymbol{\vartheta}^{\rm opt}$ and $\boldsymbol{\theta}^{\rm opt}$, such that the value function and policy are the solution to problem $\boldsymbol{\mathcal{P}}_1$.

IV. Learning Algorithm Design

We apply an actor-critic-based reinforcement learning [12], where the actor determines the policy and the critic uses the value function to evaluate the policy. Algorithm 1 describes our proposed algorithm. The actor and critic updates are performed when the new system state is observed by the load aggregator at the beginning of each time slot. Hence, the load aggregator only goes through one iteration per time slot. We use index k to refer to both iteration and time slot.

Line 1 describes the initiation phase. The loop involving Lines 2 to 13 describes the actor and critic updates and the action selection phase in iteration k. In Line 3, the load aggregator observes state $\boldsymbol{s}_k$ at the beginning of time slot k. For $k = 1$, the load aggregator does not have any experience from its past decisions. Thus, it performs the action selection phase in Lines 9 to 11 to determine an action $\boldsymbol{\varphi}_1(\boldsymbol{s}_1)$ and receive the immediate cost $c(\boldsymbol{s}_1,\, \boldsymbol{\varphi}_1(\boldsymbol{s}_1))$. For iteration $k > 1$, the load aggregator performs the actor and critic updates phase

Algorithm 1 Load Control Algorithm.

```
1: Set $k := 1$, $\varepsilon := 10^{-5}$, and randomly initialize $\boldsymbol{\theta}_1$ and $\boldsymbol{\vartheta}_1$.
2: Repeat
3:    Observe current state $\boldsymbol{s}_k := (\boldsymbol{s}_{n,k},\, n \in \mathcal{N}^-,\, Y_k,\, \rho_k)$.
4:    If $k \neq 1$,
5:       Determine the TD error $\delta_{k-1}(\boldsymbol{\vartheta}_{k-1})$ according to (8).
6:       Determine the updated vector $\boldsymbol{\vartheta}_k$ according to (9).
7:       Determine the updated vector $\boldsymbol{\theta}_k$ according to (10).
8:    End if
9:    Use policy $\widetilde{\boldsymbol{\pi}}_k(\boldsymbol{s}_k,\, \boldsymbol{\theta}_k)$ to select a vector $\boldsymbol{P}^{\rm c}_k(\boldsymbol{s}_k) \in \Phi^{\rm c}_k(\boldsymbol{s}_k)$.
10:   Solve optimization problem $\boldsymbol{\mathcal{P}}_3$ to obtain a feasible action
      vector $\boldsymbol{\varphi}_k(\boldsymbol{s}_k) = (\widehat{\boldsymbol{P}}^{\rm c}_k(\boldsymbol{s}_k),\, \mathbf{W}_k(\boldsymbol{s}_k))$.
11:   Receive the immediate cost $c(\boldsymbol{s}_k,\, \boldsymbol{\varphi}_k(\boldsymbol{s}_k))$ for action $\boldsymbol{\varphi}_k(\boldsymbol{s}_k)$.
12:   $k := k + 1$.
13: Until $||\boldsymbol{\vartheta}_{k-1} - \boldsymbol{\vartheta}_{k-2}|| < \varepsilon$ and $||\boldsymbol{\theta}_{k-1} - \boldsymbol{\theta}_{k-2}|| < \varepsilon,\ k > 2$.
```

in Lines 5 to 7. In Line 5, the load aggregator computes the temporal difference (TD) error $\delta_{k-1}(\boldsymbol{\vartheta}_{k-1})$ corresponding to the previous iteration $k-1$ as follows:

$$\begin{aligned}\delta_{k-1}(\boldsymbol{\vartheta}_{k-1}) = {} & c(\boldsymbol{s}_{k-1},\, \boldsymbol{\varphi}_{k-1}(\boldsymbol{s}_{k-1})) + \beta V^{\boldsymbol{\pi}(\boldsymbol{\theta}_{k-1})}(\boldsymbol{s}_k,\, \boldsymbol{\vartheta}_{k-1}) \\ & - V^{\boldsymbol{\pi}(\boldsymbol{\theta}_{k-1})}(\boldsymbol{s}_{k-1},\, \boldsymbol{\vartheta}_{k-1}). \end{aligned} \tag{8}$$

In (8), the value function in states $\boldsymbol{s}_k$ and $\boldsymbol{s}_{k-1}$ can be obtained using *forward propagation* in the neural network for the value function under the given vector $\boldsymbol{\vartheta}_{k-1}$ [13]. In Line 6, the load aggregator performs the critic update to obtain the updated parameter vector $\boldsymbol{\vartheta}_k$ using the following rule:

$$\boldsymbol{\vartheta}_k = \boldsymbol{\vartheta}_{k-1} - \alpha^{\rm c}_{k-1} \nabla_{\boldsymbol{\vartheta}}\, \delta^2_{k-1}(\boldsymbol{\vartheta})\big|_{\boldsymbol{\vartheta}=\boldsymbol{\vartheta}_{k-1}}, \tag{9}$$

where $\alpha^{\rm c}_{k-1}$ is the step size for the critic update in iteration k. The gradient vector $\nabla_{\boldsymbol{\vartheta}}\, \delta^2_{k-1}(\boldsymbol{\vartheta})$ is obtained using the *back propagation* algorithm [13] in the neural network associated with the value function. In Line 7, the load aggregator performs the actor update to determine the updated parameter vector $\boldsymbol{\theta}_k$ using the following update rule:

$$\begin{aligned}\boldsymbol{\theta}_k = {} & \boldsymbol{\theta}_{k-1} - \alpha^{\rm a}_{k-1}\, \delta_{k-1}(\boldsymbol{\vartheta}_{k-1}) \\ & \times \nabla_{\boldsymbol{\theta}}\big(-\ln\big(\widetilde{\pi}(\boldsymbol{s}_{k-1}, \boldsymbol{P}^{\rm c}_{k-1}(\boldsymbol{s}_{k-1}), \boldsymbol{\theta})\big)\big)\big|_{\boldsymbol{\theta}=\boldsymbol{\theta}_{k-1}}, \end{aligned} \tag{10}$$

where $\alpha^{\rm a}_{k-1}$ is the step size for the actor update in iteration k. The gradient with respect to $\boldsymbol{\theta}$ is obtained using the back propagation algorithm. Lines 9 to 11 describe the action selection phase. In Line 9, the load aggregator selects vector $\boldsymbol{P}^{\rm c}_k(\boldsymbol{s}_k) \in \Phi^{\rm c}_k(\boldsymbol{s}_k)$. In Line 10, it solves problem $\boldsymbol{\mathcal{P}}_3$ to obtain the feasible action $\boldsymbol{\varphi}_k(\boldsymbol{s}_k) = (\widehat{\boldsymbol{P}}^{\rm c}_k(\boldsymbol{s}_k),\, \mathbf{W}_k(\boldsymbol{s}_k))$. In Line 11, the load aggregator broadcasts action $\boldsymbol{\varphi}_k(\boldsymbol{s}_k)$ to the ECCs and computes the cost $c(\boldsymbol{s}_k,\, \boldsymbol{\varphi}_k(\boldsymbol{s}_k))$ using (4). Next time slot is started in Line 12. In Line 13, the stopping criterion is given. For Algorithm 1 to converge to the solution of $\mathcal{P}_1$, it is necessary that $\alpha^{\rm a}_k$ and $\alpha^{\rm c}_k$ are diminishing step sizes with $\sum_{k=1}^{\infty}\alpha^{\rm a}_k = \sum_{k=1}^{\infty}\alpha^{\rm c}_k = \infty$ and $\sum_{k=1}^{\infty}(\alpha^{\rm a}_k)^2 = \sum_{k=1}^{\infty}(\alpha^{\rm c}_k)^2 < \infty$, and $\sum_{k=1}^{\infty}(\alpha^{\rm a}_k/\alpha^{\rm c}_k)^d < \infty$ for some $d > 0$ [12].

V. Performance Evaluation

In this section, we evaluate the performance of the proposed load control algorithm on an IEEE 85-bus distribution feeder with 59 households. The network data can be found in [16]. The lower and upper bounds for bus voltage magnitudes are 0.9 pu and 1.1 pu, respectively. The maximum apparent power flow through the transmission lines is 1.05 pu. Unless stated otherwise, no limit is considered for the active and reactive power flow in the substation bus. One day is divided into 96 time slots with duration of 15 minutes. We consider a real-time pricing generated from a truncated normal distribution with the average values shown in Fig. 1 and a standard deviation of 0.5 cents per kW. The random pricing scheme can be modeled as an MDP. To obtain the MDP for the base load $P^{\rm b}_n(t)$, and parameters $P^{\rm c,min}_n(t)$ and $P^{\rm c,des}_n(t)$, $t \in \mathcal{T}$, $n \in \mathcal{N}$, we use the state model in [14] by considering six controllable and six uncontrollable appliances for each household. The discount factor β is set to 0.9. The discomfort cost for a household is chosen at random from the interval [0.5, 2] cents per time slot between 3 pm and 6 am, and is set to 10 cents per time slot otherwise. For the actor, we consider a neural network with five hidden layers and 649 nodes in each layer (11 nodes per household). For the critic, we consider a neural network with five hidden layers and 295 nodes in each layer (5 nodes per household). We use leaky rectified linear unit activation functions. The step sizes are set to $\alpha^{\rm a}_k = 25/k$ and $\alpha^{\rm c}_k = 100/k^{0.6}$. We perform simulations using MATLAB/CVX with MOSEK solver and PYTORCH library in PYTHON 3.7.

We first show how Algorithm 1 enables the load aggregator reducing the peak load. For the sake of comparison, we consider four load profiles: *i*) desirable demand without load control, *ii*) scheduled demand with load control and learning using Algorithm 1, *iii*) desirable demand with load control, and *iv*) scheduled demand with load control but without learning (i.e., untrained neural networks). Fig. 2(a) shows the load profile of household 1 in day 30 as an example. By using Algorithm 1, the peak load is reduced from 0.9 kW to 0.7 kW (i.e., 23% reduction). Without learning, however, the load aggregator chooses an action at random, and hence the load control does not have an acceptable performance. Reducing the controllable load in a time slot causes the desirable demand increases in upcoming time slots. This can be interpreted as shifting the load demand from one time slot to future time slots. Fig. 2(b) shows the aggregate load demand of all households in day 30. Peak load reduction by 16.7% (from 54 kW to 45 kW) is achieved using Algorithm 1.

Next we show that considering the distribution network constraints is necessary, as it can affect the load aggregator's decision making. We consider an upper limit of 20 kW for the active power injection into the substation from 12 am to 6 am. Fig. 3 shows that the supply limit affects the learning of the load aggregator, resulting in a higher demand during the period from 3 pm to 12 am to avoid shifting too much load demand to time period 12 am to 6 am. Also, the load demand increases slightly between 6 am and 12 pm of the next day as a result of shifting the control load demands. Results show that power networks constraints can affect the feasible action space, and hence the load control policy of the load aggregator.

Finally, we study the convergence of Algorithm 1. Fig. 4(a) shows the convergence of TD error during the first 30 days (i.e., 2880 iterations). The joint goal of critic and actor updates in (9) and (10) is to decrease the TD error and make its expectation to be zero. Also, Fig. 4(b) shows that the value

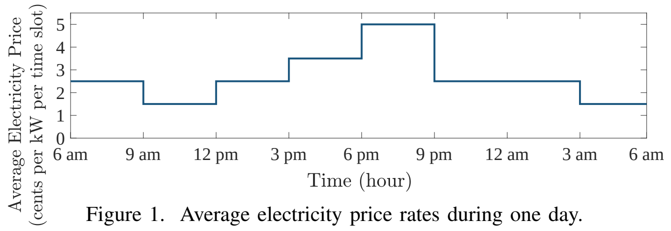

Figure 1. Average electricity price rates during one day.

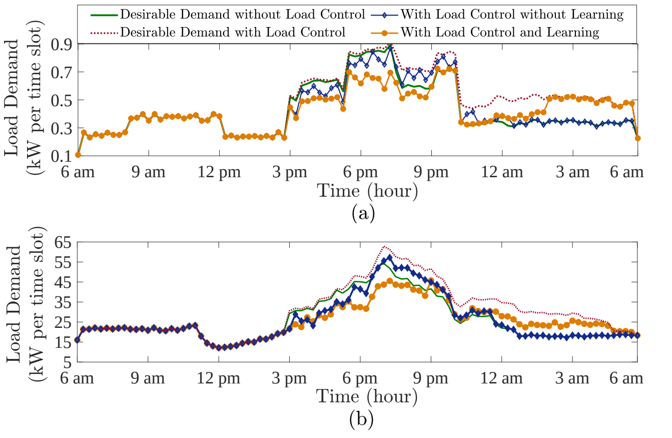

Figure 2. (a) Load demand of household 1; (b) Aggregate demand of 59 households in the feeder in day 30.

function decreases gradually and converges from \$15 to \$11 per time slot (i.e., 26.6% reduction). That is, the policy and value function converge to the optimal solution of problem $\mathcal{P}_1$ at the same time. Notice that Algorithm 1 can converge to a near optimal solution in the first 10 days (in about 1000 iterations). Hence, Algorithm 1 is applicable in practice to determine the optimal policy and value function.

VI. Concluding Remarks

In this paper, we studied the load control problem for a load aggregator in a distribution network. We deployed deep reinforcement learning approach to enable load management under uncertainty in the electricity price, load demand, and customers' discomfort cost. To address the non-convexity of power flow constraints, we transform the problem of updating neural network parameters into a sequence of SDPs. By simulations, we showed that the load aggregator can benefit from 16.7% reduction in the aggregate load demand during peak hours. A customer also can benefit from 26.6% reduction in its expected cost. The proposed learning algorithm can converge to the optimal solution in an acceptable number of iterations, which is equivalent to a few days in practical applications. For future work, we will study a distributed algorithm design for the households' appliances scheduling.

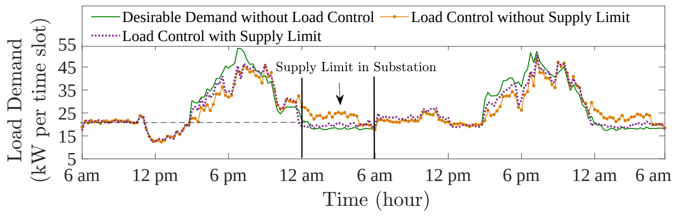

Figure 3. Aggregate demand in day 30 with and without limited injected active power into the substation.

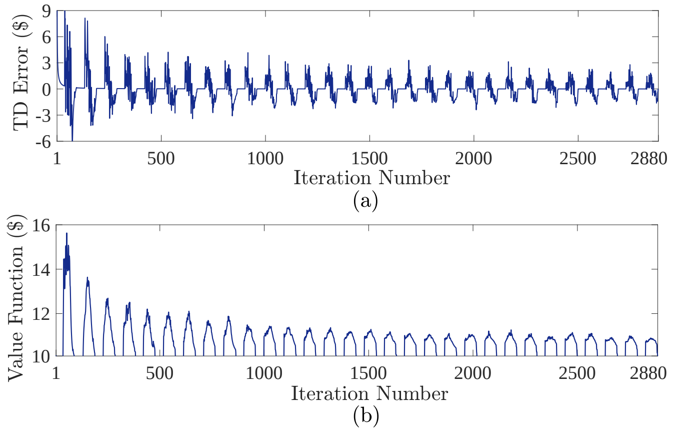

Figure 4. (a) TD error; (b) Value function versus iteration number k.

References

[1] J. S. Vardakas, N. Zorba, and C. V. Verikoukis, "A survey on demand response programs in smart grids: Pricing methods and optimization algorithms," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 1, pp. 152–178, Firstquarter 2015.

[2] X. Yang, Y. Zhang, H. He, S. Ren, and G. Weng, "Real-time demand side management for a microgrid considering uncertainties," *IEEE Trans. on Smart Grid*, vol. 10, no. 3, pp. 3401–3414, May 2019.

[3] K. Garifi, K. Baker, B. Touri, and D. Christensen, "Stochastic model predictive control for demand response in a home energy management system," in *Proc. of IEEE Power Energy Society General Meeting (PESGM)*, Portland, OR, Aug. 2018.

[4] Y. F. Du, L. Jiang, Y. Li, and Q. Wu, "A robust optimization approach for demand side scheduling considering uncertainty of manually operated appliances," *IEEE Trans. on Smart Grid*, vol. 9, no. 2, pp. 743–755, Mar. 2018.

[5] M. Zuniga Alvarez, K. Agbossou, A. Cardenas, S. Kelouwani, and L. Boulon, "Demand response strategy applied to residential electric water heaters using dynamic programming and K-means clustering," accepted for publication in *IEEE Trans. Sustain. Energy*, Feb. 2019.

[6] Y. Du and F. Li, "Intelligent multi-microgrid energy management based on deep neural network and model-free reinforcement learning," accepted for publication in *IEEE Trans. on Smart Grid*, Jul. 2019.

[7] E. Mocanu, D. C. Mocanu, P. H. Nguyen, A. Liotta, M. E. Webber, M. Gibescu, and J. G. Slootweg, "On-line building energy optimization using deep reinforcement learning," *IEEE Trans. on Smart Grid*, vol. 10, no. 4, pp. 3698–3708, Jul. 2019.

[8] T. Wei, Yanzhi Wang, and Q. Zhu, "Deep reinforcement learning for building HVAC control," in *Proc. of ACM/EDAC/IEEE Design Automation Conf. (DAC)*, Austin, TX, Jun. 2017.

[9] B. J. Claessens, P. Vrancx, and F. Ruelens, "Convolutional neural networks for automatic state-time feature extraction in reinforcement learning applied to residential load control," *IEEE Trans. on Smart Grid*, vol. 9, no. 4, pp. 3259–3269, Jul. 2018.

[10] K. L. López, C. Gagné, and M. Gardner, "Demand-side management using deep learning for smart charging of electric vehicles," *IEEE Trans. on Smart Grid*, vol. 10, no. 3, pp. 2683–2691, May 2019.

[11] Z. Wan, H. Li, H. He, and D. Prokhorov, "Model-free real-time EV charging scheduling based on deep reinforcement learning," *IEEE Trans. on Smart Grid*, vol. 10, no. 5, pp. 5246–5257, Sept. 2019.

[12] I. Grondman, L. Busoniu, G. A. D. Lopes, and R. Babuska, "A survey of actor-critic reinforcement learning: Standard and natural policy gradients," *IEEE Trans. on Systems, Man, and Cybernetics*, vol. 42, no. 6, pp. 1291–1307, Nov. 2012.

[13] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *Proc. of Int'l. Conf. on Machine Learning*, NYC, NY, Jun. 2016.

[14] S. Bahrami, V.W.S.Wong, and J. Huang, "An online learning algorithm for demand response in smart grid," *IEEE Trans. on Smart Grid*, vol. 9, no. 5, pp. 4712–4725, Sept. 2018.

[15] J. Lavaei and S. H. Low, "Zero duality gap in optimal power flow problem," *IEEE Trans. on Power Systems*, vol. 27, pp. 92–107, Feb. 2012.

[16] R. D. Zimmerman and C. E. Murillo-Sanchez, "MATPOWER (Version 7.0)," 2019. [Online]. Available: https://matpower.org.