Optimal Tap Setting of Voltage Regulation Transformers Using Batch Reinforcement Learning

Hanchen Xu, Alejandro D. Domínguez-García, *Member, IEEE* and Peter W. Sauer, *Life Fellow, IEEE*

***Abstract*—In this paper, we address the problem of setting the tap positions of load tap changers (LTCs) for voltage regulation in power distribution systems. The objective is to find a policy that maps measurements of voltage magnitudes and topology information to LTC tap ratio changes so as to minimize the voltage deviation across the system. We formulate this problem as a Markov decision process (MDP), and propose a data and computationally efficient batch reinforcement learning (RL) algorithm to solve it. To circumvent the "curse of dimensionality" resulting from the large state and action spaces, we propose a sequential learning algorithm to learn an action-value function for each LTC, based on which the optimal tap positions can be directly determined. By taking advantage of a linearized power flow model, we propose an algorithm to estimate the voltage magnitudes under different tap settings, which allows the RL algorithm to explore the state and action spaces freely offline without impacting the system operation. The effectiveness of the proposed algorithm is validated via numerical simulations on the IEEE 13-bus and 123-bus distribution test feeders.**

***Index Terms*—voltage regulation, load tap changer, data-driven, Markov decision process, reinforcement learning.**

I. Introduction

VOLTAGE regulation transformers—also referred to as load tap changers (LTCs)—are widely utilized in power distribution systems to regulate the voltage magnitudes along a feeder. Conventionally, the tap position of each LTC is controlled through automatic voltage control (AVC) based on local voltage measurements [1]. However, this simple scheme may no longer be suitable in the presence of bidirectional power flows induced by the integration of renewable energy resources [2]. In addition, the AVC-based scheme does not take into account optimality in any sense; particularly, the voltage deviation from some reference value may not be minimized.

Alternatively, transformer tap positions can also be optimized jointly with active and reactive power generation by solving an optimal power flow (OPF) problem [3]–[7]. In OPF-based approaches, the key focus is on the formulation of an OPF with LTCs that have discrete tap positions and the development of efficient algorithms that solve the OPF. For example, in [3], the authors modeled the tap ratios as continuous variables and included a penalty term in the objective function that forces these continuous variables representing tap ratios to stay close to their feasible discrete values; then, the OPF problem contains only continuous variables and thus can be solved efficiently. The authors in [5] also modeled LTC tap ratios as continuous variables in the OPF, and developed a strategy to round off the continuous variables to their feasible discrete values. In [6], the authors cast the optimal tap setting problem of LTCs as a rank-constrained semidefinite program that is further relaxed by dropping the rank-one constraint, which avoids the non-convexity and integer variables, and thus, the problem can be solved efficiently.

OPF-based approaches typically deal with one snapshot of system conditions; consequently, the optimal tap setting problem needs to be solved for each snapshot in real time, which is not desirable when the computation involved in solving the OPF is non-trivial. In addition, these approaches require complete system knowledge including active and reactive power injections, and transmission/distribution line parameters, as well as system topology, for every snapshot. While it may be reasonable to assume that such information is available for transmission systems, the situation in distribution systems is quite different. Accurate line parameters may not be known and power injections at each bus may not be available in real time, which prevents the application of these approaches [8].

Different from OPF-based approaches, the objective of this paper is to develop a methodology which learns a control policy that maps a set of measurements to tap ratio changes of LTCs in a power distribution system so that voltage deviations from a reference value are minimized. The learned control policy is expected to find the optimal tap positions with the minimal information and computation time when executed online. To this end, we first cast the optimal tap setting problem as a multi-stage decision making problem and formulate it as a Markov decision process (MDP), where the actions are the discrete tap ratio changes of the LTCs and the states are constructed from available measurements. MDPs can be solved effectively using reinforcement learning (RL) algorithms, as has been demonstrated in a variety of applications such as power system stability control [9], reactive power control [10], [11], demand response [12], [13], economic dispatch [14], and electricity markets [15], [16]. Readers are referred to [17] for a more comprehensive review on the application of RL to power system problems. Particularly, regarding voltage regulation, authors in [10] applied the classical Q learning algorithm to solve the voltage control problem in power transmission systems, and authors in [11] proposed a distributed version of the Q learning algorithm to solve the voltage control problem. Yet, these approaches require the discretization of the state space. The discretized state space will better approximate the original state space if a small step size is used in the discretization process, yet the resulting MDP will face the "curse of dimensionality" that makes the computation of the optimal policy intractable [18]. A large step size can alleviate

The authors are with the Department of Electrical and Computer Engineering at the University of Illinois at Urbana-Champaign, Urbana, IL 61801, USA. Email: {hxu45, aledan, psauer}@illinois.edu.

the computational burden caused by the high dimensionality of the state space, but at the cost of potentially degrading performance significantly. Moreover, complete power system models are required in order to train these algorithms.

In this paper, we will develop an algorithm that circumvents the "curse of dimensionality" faced by the infinite state space as well as the finite action space, the dimension of which increases exponentially as the number of LTCs grows. To this end, we propose a data and computationally efficient batch RL algorithm—the least squares policy iteration (LSPI) based sequential learning algorithm—to learn an action-value function sequentially for each LTC. Once the learning of the action-value function is completed, we can determine the policy for optimally setting the LTC taps. We emphasize that the optimal policy can be learned offline by the RL algorithm, where most computational burden takes place. However, when executed online, the required computation to find the optimal tap positions is minimal.

Another challenge we will tackle is the acquisition of adequate transition samples that sufficiently explore the MDP state and action spaces. In practice, it is hard to obtain such samples in real power systems since this requires changing tap settings and other controls to excite the system and record voltage responses, which may jeopardize system operational reliability and incur economic costs. Since accurate models for power distribution systems may not be available, acquiring transition samples from a complete system simulation model may also be infeasible. To resolve this issue, we take advantage of a linearized power flow model and develop a virtual transition generator, which can estimate voltage magnitudes under different tap settings so that the state and action spaces can be explored freely offline without impacting the real system. The proposed voltage magnitude estimation algorithm requires only voltage magnitude measurements and system topology information, yet without any information on power injections or line parameters.

The major contributions of this paper are as follows:

1) the formulation of the optimal tap setting problem of LTCs as an MDP, and the development of a data and computationally efficient sequential batch RL algorithm that solves this MDP without facing the "curse of dimensionality";
2) the development of a virtual transition generator based on historical data that enables the exploration of the state and action spaces without impacting the real system or requiring a complete system model;
3) the illustration of the application of the proposed algorithm on two IEEE distribution test feeders through simulations.

The remainder of the paper is organized as follows. A linearized power flow model that includes the effect of LTCs and the optimal tap setting problem are presented in Section II. A primer on MDPs and the LSPI algorithm is provided in Section III. An MDP formulation for the optimal tap setting problem and the batch RL algorithm that solves this MDP are presented in Sections V and IV, respectively. Numerical simulation results on two IEEE test feeders are presented in Section VI. Concluding remarks are provided in Section VII.

II. Preliminaries

In this section, we review a linearized power flow model for power distribution systems, and modify it to include the effect of LTCs. We also describe the LTC tap setting problem.

A. Power Distribution System Model

Consider a balanced radial power distribution system that consists of a set of buses indexed by the elements in $\mathcal{N} = \{0, 1, \cdots, N\}$, and a set of transmission lines indexed by the elements in $\mathcal{L} = \{1, \cdots, L\}$. Each line $\ell \in \mathcal{L}$ is associated with an ordered pair $(i, j) \in \mathcal{N} \times \mathcal{N}$. Assume bus 0 is an ideal voltage source that corresponds to a substation bus, which is the only connection of the distribution system to the bulk power grid.

Let V_i denote the magnitude of the voltage at bus i, $i \in \mathcal{N}$, and define $v_i := V_i^2$; note that v_0 is a constant since bus 0 is assumed to be an ideal voltage source. Let p_i and q_i denote the active power injection and reactive power injection at bus i, $i \in \mathcal{N}$, respectively. For each line $\ell \in \mathcal{L}$ that is associated with (i, j), let p_{ij} and q_{ij} respectively denote active and reactive power flows on line (i, j), which are positive if the flow of power is from bus i to bus j and negative otherwise. Let r_ℓ and x_ℓ denote the resistance and reactance of line ℓ, $\ell \in \mathcal{L}$. The relation between squared voltage magnitudes, power injections, and line power flows, can be captured by the so-called LinDisfFlow model [19] as follows:

$$p_{ij} = -p_j + \sum_{k:(j,k)\in\mathcal{L}} p_{jk}, \tag{1a}$$

$$q_{ij} = -q_j + \sum_{k:(j,k)\in\mathcal{L}} q_{jk}, \tag{1b}$$

$$v_i - v_j = 2(r_\ell p_{ij} + x_\ell q_{ij}), \tag{1c}$$

where ℓ is associated with (i, j).

Define $\boldsymbol{r} = [r_1, \cdots, r_L]^\top$ and $\boldsymbol{x} = [x_1, \cdots, x_L]^\top$. Let $\tilde{\boldsymbol{M}} = [\tilde{M}_{i\ell}] \in^{(N+1)\times L}$, with $\tilde{M}_{i\ell} = 1$ and $\tilde{M}_{j\ell} = -1$ if line ℓ is associated with (i, j), and all other entries equal to zero. Let $\boldsymbol{m}^\top$ denote the first row of $\tilde{\boldsymbol{M}}$ and $\boldsymbol{M}$ the matrix that results by removing $\boldsymbol{m}^\top$ from $\tilde{\boldsymbol{M}}$. For a radial distribution system, $L = N$, and $\boldsymbol{M}$ is invertible. Define $\boldsymbol{v} = [v_1, \cdots, v_N]^\top$, $\boldsymbol{p} = [p_1, \cdots, p_N]^\top$, and $\boldsymbol{q} = [q_1, \cdots, q_N]^\top$. Then, the LinDistFlow model in (1) can be written as follows:

$$\boldsymbol{M}^\top \boldsymbol{v} + \boldsymbol{m} v_0 = 2\mathrm{diag}(\boldsymbol{r})\boldsymbol{M}^{-1}\boldsymbol{p} + 2\mathrm{diag}(\boldsymbol{x})\boldsymbol{M}^{-1}\boldsymbol{q}, \tag{2}$$

where $\mathrm{diag}(\cdot)$ returns a diagonal matrix with the entries of the argument as its diagonal elements.

The standard model for an LTC in the literature is shown in Fig. 1 (see, e.g., [1]), where $\mathrm{i} = \sqrt{-1}$, line ℓ is associated with (i, j), and t_l is the tap ratio of LTC l. Typically, the tap ratio can possibly take on 33 discrete values ranging from 0.9 to 1.1, by an increment of $5/8\%$ p.u., i.e., $t_l \in \mathcal{T} = \{0.9, 0.90625, \cdots, 1.09375, 1.1\}$ [1]. We index the 33 tap positions by $-16, \cdots, -1, 0, 1, \cdots, 16$ for convenience. Let $\Delta\mathcal{T} = \{0, \pm 0.00625, \cdots, \pm 0.19375, \pm 0.2\}$ denote the set of possible LTC tap ratio changes between two consecutive time instants. Denote the tap ratio change of LTC l by Δt_l. Note that the set of feasible values of Δt_l at any time instant

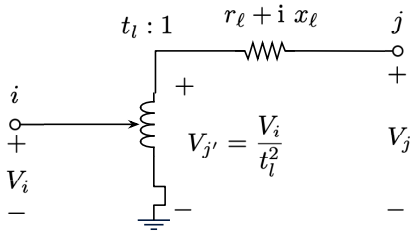

Fig. 1. Load tap changer model.

depends on the current tap ratio of the LTC, t_l, and is a subset of $\Delta\mathcal{T}$. As such, we denote the set of feasible values of Δt_l by $\Delta\mathcal{T}(t_l) \subset \Delta\mathcal{T}$.

Let $\mathcal{L}^t = \{1, \cdots, L^t\}$ denote the index set of LTCs. For LTC $l \in \mathcal{L}^t$ on line ℓ that is associated with (i, j), the voltage relation in the LinDistFlow model, i.e., (1c), needs to be modified as follows:

$$\frac{1}{t_l^2} v_i - v_j = 2(r_\ell p_{ij} + x_\ell q_{ij}). \tag{3}$$

Define $\boldsymbol{t} = [t_1, \cdots, t_{L^t}]^\top$. Let $\tilde{\boldsymbol{M}}(\boldsymbol{t}) = [\tilde{M}_{i\ell}(\boldsymbol{t})] \in \mathbb{R}^{(N+1)\times L}$, with $\tilde{M}_{i\ell}(\boldsymbol{t}) = 1$ and $\tilde{M}_{j\ell}(\boldsymbol{t}) = -1$ if there is no LTC on line ℓ, $\tilde{M}_{i\ell}(\boldsymbol{t}) = \frac{1}{t_l^2}$ and $\tilde{M}_{j\ell}(\boldsymbol{t}) = -1$ if LTC l is on line ℓ, and all other entries equal to zero. Let $\boldsymbol{m}(\boldsymbol{t})^\top$ denote the first row of $\tilde{\boldsymbol{M}}(\boldsymbol{t})$ and $\boldsymbol{M}(\boldsymbol{t})$ the matrix resulting from removing $\boldsymbol{m}(\boldsymbol{t})^\top$ from $\tilde{\boldsymbol{M}}(\boldsymbol{t})$. The matrix $\boldsymbol{M}(\boldsymbol{t})$ is non-singular when the power distribution system is connected. Then, the modified LinDistFlow model that takes into account the LTCs is given by:

$$\boldsymbol{M}(\boldsymbol{t})^\top \boldsymbol{v} + \boldsymbol{m}(\boldsymbol{t}) v_0 = 2\mathrm{diag}(\boldsymbol{r})\boldsymbol{M}^{-1}\boldsymbol{p} + 2\mathrm{diag}(\boldsymbol{x})\boldsymbol{M}^{-1}\boldsymbol{q}. \tag{4}$$

B. Optimal Tap Setting Problem

To effectively regulate the voltages in a power distribution system, the tap positions of LTCs need to be set appropriately. The objective of the optimal tap setting problem is to find a policy $\boldsymbol{\pi}$ that determines the tap ratio changes of the LTCs so as to minimize the voltage deviation from some reference value, denoted by $\boldsymbol{v}^\star$, based on measurements of current tap ratios and voltage magnitudes, i.e., $\boldsymbol{\pi} : (\boldsymbol{t}, \boldsymbol{v}) \to \Delta\boldsymbol{t}$, where $\boldsymbol{t} \in \mathcal{T}^{L^t}$, $\boldsymbol{v} \in \mathbb{R}^N$, $\Delta\boldsymbol{t} = [\Delta t_1, \cdots, \Delta t_{L^t}]^\top$, and $\Delta t_l \in \Delta\mathcal{T}(t_l)$ for $l \in \mathcal{L}^t$. Throughout this paper, we make the following two assumptions:

A1. The distribution system topology is known but the line parameters are unknown.

A2. The active and reactive power injections are not measured and their probability distributions are unknown.

III. Markov Decision Process and Batch Reinforcement Learning

In this section, we provide some background on MDPs and the batch RL algorithm, a type of data and computationally efficient and stable algorithm for solving MDPs with unknown transition models.

A. Markov Decision Process

An MDP is defined as a 5-tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$, where $\mathcal{S}$ is a finite set of states; $\mathcal{A}$ is a finite set of actions; $\mathcal{P}$ is a Markovian transition model that denotes the probability of transitioning from one state into another after taking an action; $\mathcal{R} : \mathcal{S}\times\mathcal{A}\times\mathcal{S} \to \mathbb{R}$ is a reward function such that, for $\boldsymbol{s}, \boldsymbol{s}' \in \mathcal{S}$ and $\boldsymbol{a} \in \mathcal{A}$, $r = \mathcal{R}(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{s}')$ is the reward obtained when the system transitions from state $\boldsymbol{s}$ into state $\boldsymbol{s}'$ after taking action $\boldsymbol{a}$; and $\gamma \in [0, 1)$ is a discount factor (see, e.g., [20]).[1] We refer to the 4-tuple $(\boldsymbol{s}, \boldsymbol{a}, r, \boldsymbol{s}')$, where $\boldsymbol{s}'$ is the state following $\boldsymbol{s}$ after taking action $\boldsymbol{a}$ and $r = \mathcal{R}(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{s}')$, as a transition.

Let $\boldsymbol{S}_k$ and $\boldsymbol{A}_k$ denote the state and action at time instant k, respectively, and R_k the reward received after taking action $\boldsymbol{A}_k$ in state $\boldsymbol{S}_k$. Let $\mathbb{P}$ denote the probability operator; then, $\mathcal{P}_k(\boldsymbol{s}'|\boldsymbol{s}, \boldsymbol{a}) := \mathbb{P}\{\boldsymbol{S}_{k+1} = \boldsymbol{s}' | \boldsymbol{S}_k = \boldsymbol{s}, \boldsymbol{A}_k = \boldsymbol{a}\}$ is the probability of transitioning from state $\boldsymbol{s}$ into state $\boldsymbol{s}'$ after taking action $\boldsymbol{a}$ at instant k. Throughout this paper, we assume time-homogeneous transition probabilities, hence we drop the subindex k and just write $\mathcal{P}(\boldsymbol{s}'|\boldsymbol{s}, \boldsymbol{a})$.

Let $\bar{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ denote the expected reward for a state-action pair $(\boldsymbol{s}, \boldsymbol{a})$; then, we have

$$\bar{R}(\boldsymbol{s}, \boldsymbol{a}) = \mathbb{E}[R|\boldsymbol{s}, \boldsymbol{a}] = \sum_{\boldsymbol{s}' \in \mathcal{S}} \mathcal{R}(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{s}')\mathcal{P}(\boldsymbol{s}'|\boldsymbol{s}, \boldsymbol{a}), \tag{5}$$

where $\mathbb{E}[\cdot]$ denotes the expectation operation. The total discounted reward from time instant k and onwards, denoted by G_k, also referred to as the return, is given by

$$G_k = \sum_{k'=k}^{\infty} \gamma^{k'-k} R_{k'}. \tag{6}$$

A deterministic policy $\boldsymbol{\pi}$ is a mapping from $\mathcal{S}$ to $\mathcal{A}$, i.e., $\boldsymbol{a} = \boldsymbol{\pi}(\boldsymbol{s}), \boldsymbol{s} \in \mathcal{S}, \boldsymbol{a} \in \mathcal{A}$. The action-value function under policy $\boldsymbol{\pi}$ is defined as follows:

$$Q^{\boldsymbol{\pi}}(\boldsymbol{s}, \boldsymbol{a}) = \mathbb{E}[G_k | \boldsymbol{S}_k = \boldsymbol{s}, \boldsymbol{A}_k = \boldsymbol{a}; \boldsymbol{\pi}], \tag{7}$$

which is the expected return when taking action $\boldsymbol{a}$ in state $\boldsymbol{s}$, and following policy $\boldsymbol{\pi}$ afterwards. Intuitively, the action-value function quantifies, for a given policy π, how "good" the state-action pair $(\boldsymbol{s}, \boldsymbol{a})$ is in the long run.

Let $Q^*(\cdot, \cdot)$ denote the optimal action-value function—the maximum action-value function over all policies, i.e., $Q^*(\boldsymbol{s}, \boldsymbol{a}) = \max_{\boldsymbol{\pi}} Q^{\boldsymbol{\pi}}(\boldsymbol{s}, \boldsymbol{a})$. All optimal policies share the same optimal action-value function. Also, the greedy policy with respect to $Q^*(\boldsymbol{s}, \boldsymbol{a})$, i.e., $\boldsymbol{\pi}^*(\boldsymbol{s}) = \arg\max_{\boldsymbol{a}} Q^*(\boldsymbol{s}, \boldsymbol{a})$ is an optimal policy. Then, it follows from (6) and (7) that $Q^*(\boldsymbol{s}, \boldsymbol{a})$ satisfies the following Bellman optimality equation (see, e.g., [18]):

$$Q^*(\boldsymbol{s}, \boldsymbol{a}) = \bar{R}(\boldsymbol{s}, \boldsymbol{a}) + \gamma \sum_{\boldsymbol{s}' \in \mathcal{S}} \mathcal{P}(\boldsymbol{s}'|\boldsymbol{s}, \boldsymbol{a}) \max_{\boldsymbol{a}' \in \mathcal{A}} Q^*(\boldsymbol{s}', \boldsymbol{a}'). \tag{8}$$

The MDP is solved if we find $Q^*(\boldsymbol{s}, \boldsymbol{a})$, and correspondingly, the optimal policy $\boldsymbol{\pi}^*$. It is important to emphasize that (8) is

[1] These definitions can be directly extended to the case where the the set of states is infinite. Due to space limitation, this case is not discussed in detail here.

key in solving the MDP. For ease of notation, in the rest of this paper, we simply write the $Q^*(\boldsymbol{s},\boldsymbol{a})$ as $Q(\boldsymbol{s},\boldsymbol{a})$.

When both the state and the action sets are finite, the action-value function can be exactly represented in a tabular form that covers all possible pairs $(\boldsymbol{s},\boldsymbol{a}) \in \mathcal{S}\times\mathcal{A}$. In this case, if $\mathcal{P}$ is also known, then the MDP can be solved using, e.g., the so-called policy iteration and value iteration algorithms (see, e.g., [18]). If $\mathcal{P}$ is unknown but samples of transitions are available, the MDP can be solved by using RL algorithms such as the Q-learning algorithm (see, e.g., [21]).

B. Batch Reinforcement Learning

When $\mathcal{S}$ is not finite, conventional Q-learning based approaches require discretization of $\mathcal{S}$ (see, e.g., [10] and [11]), which may incur the "curse of dimensionality." More practically, when the number of elements in $\mathcal{S}$ is large or $\mathcal{S}$ is not finite, the action-value function can be approximated by some parametric functions such as linear functions [20] and neural networks [22], or non-parametric functions such as decision trees [23]. Let $\hat{Q}(\cdot,\cdot)$ denote the approximate optimal action-value function. Using a linear function approximation, $\hat{Q}(\boldsymbol{s},\boldsymbol{a})$ can be represented as follows:

$$\hat{Q}(\boldsymbol{s},\boldsymbol{a}) = \boldsymbol{w}^\top \boldsymbol{\phi}(\boldsymbol{s},\boldsymbol{a}), \tag{9}$$

where $\boldsymbol{\phi}: \mathcal{S}\times\mathcal{A}\to\mathbb{R}^f$ is a feature mapping for $(\boldsymbol{s},\boldsymbol{a})$, which is also referred to as the basis function, and $\boldsymbol{w}\in\mathbb{R}^f$ is the parameter vector.

A class of stable, data and computationally efficient RL algorithms that can solve an MDP with function approximations are the batch RL algorithms—"batch" in the sense that a set of transition samples are utilized each time—such as the LSPI algorithm [20], which is considered to be the most efficient one in this class. We next explain the fundamental idea behind the LSPI algorithm. Let $\mathcal{D} = \{(\boldsymbol{s},\boldsymbol{a},r,\boldsymbol{s}') : \boldsymbol{s},\boldsymbol{s}'\in\mathcal{S},\boldsymbol{a}\in\mathcal{A}\}$ denote a set (batch) of transition samples obtained via observation or simulation. The LSPI algorithm finds the best $\boldsymbol{w}$ that fits the transition samples in $\mathcal{D}$ in an iterative manner. One way to explain the intuition behind the LSPI algorithm is as follows (the readers are referred to [20] for a more rigorous development). Define

$$g(\boldsymbol{w}) = \sum_{(\boldsymbol{s},\boldsymbol{a},r,\boldsymbol{s}')\in\mathcal{D}} \left(Q(\boldsymbol{s},\boldsymbol{a}) - \boldsymbol{w}^\top\boldsymbol{\phi}(\boldsymbol{s},\boldsymbol{a})\right)^2. \tag{10}$$

Let $\boldsymbol{w}_i$ denote the value of $\boldsymbol{w}$ that is available at the beginning of iteration i. At iteration i, the algorithm finds $\boldsymbol{w}_{i+1}$ by solving the following problem:

$$\boldsymbol{w}_{i+1} = \arg\min_{\boldsymbol{w}} g(\boldsymbol{w}), \tag{11}$$

which is an unconstrained optimization problem. The solution of (11) can be computed by setting the gradient of $g(\cdot)$ to zero as follows:

$$\frac{\partial g}{\partial \boldsymbol{w}} = -2\sum_{(\boldsymbol{s},\boldsymbol{a},r,\boldsymbol{s}')\in\mathcal{D}} \left(Q(\boldsymbol{s},\boldsymbol{a}) - \boldsymbol{w}^\top\boldsymbol{\phi}(\boldsymbol{s},\boldsymbol{a})\right)\boldsymbol{\phi}(\boldsymbol{s},\boldsymbol{a}) = \boldsymbol{0}_f. \tag{12}$$

Note that the true value of $Q(\boldsymbol{s},\boldsymbol{a})$ is not known and is substituted by the so-called temporal-difference (TD) target, $r+\gamma\boldsymbol{w}^\top\boldsymbol{\phi}(\boldsymbol{s}',\boldsymbol{a}')$, where $\boldsymbol{a}' = \arg\max_{\boldsymbol{a}\in\mathcal{A}}\boldsymbol{w}_i^\top\boldsymbol{\phi}(\boldsymbol{s}',\boldsymbol{a})$ is the optimal action in state $\boldsymbol{s}'$ determined based on $\boldsymbol{w}_i$. Note that the TD target is a sample of the right-hand-side (RHS) of (8), which serves as an estimate for the RHS of (8). We emphasize that despite $Q(\boldsymbol{s},\boldsymbol{a})$ being substituted by $r+\gamma\boldsymbol{w}^\top\boldsymbol{\phi}(\boldsymbol{s}',\boldsymbol{a}')$, the true optimal action-value function is not a function of $\boldsymbol{w}$; therefore, the gradient of g with respect to $\boldsymbol{w}$ is taken before the $Q(\boldsymbol{s},\boldsymbol{a})$ is approximated by the TD target, which does depends on $\boldsymbol{w}$. Then, after replacing $Q(\boldsymbol{s},\boldsymbol{a})$ with the TD target, (12) has the following closed-form solution:

$$\boldsymbol{w}_{i+1} = \left(\sum_{(\boldsymbol{s},\boldsymbol{a},r,\boldsymbol{s}')\in\mathcal{D}} \boldsymbol{\phi}(\boldsymbol{s},\boldsymbol{a})(\boldsymbol{\phi}(\boldsymbol{s},\boldsymbol{a}) - \gamma\boldsymbol{\phi}(\boldsymbol{s}',\boldsymbol{a}'))^\top\right)^{-1} \times \sum_{(\boldsymbol{s},\boldsymbol{a},r,\boldsymbol{s}')\in\mathcal{D}} \boldsymbol{\phi}(\boldsymbol{s},\boldsymbol{a})r. \tag{13}$$

Intuitively, at each iteration, the LSPI algorithm finds the $\boldsymbol{w}$ that minimizes the mean squared error between the TD target and $\hat{Q}(\boldsymbol{s},\boldsymbol{a})$ over all transition samples in $\mathcal{D}$. This process is repeated until change of $\boldsymbol{w}$, defined as $\|\boldsymbol{w}_{i+1}-\boldsymbol{w}_i\|$, where $\|\cdot\|$ denotes the L_2-norm, becomes smaller than a threshold ε, upon which the algorithm is considered to have converged.

The LSPI algorithm has the following three advantageous properties. First, linear functions are used to approximate the optimal action-value function, which allows the algorithm to handle MDPs with high-dimensional or continuous state spaces. Second, at each iteration, a batch of transition samples is used to update the vector $\boldsymbol{w}$ parameterizing $\hat{Q}(\cdot,\cdot)$, and these samples are reused at each iteration, thus increasing data efficiency. Third, the optimal parameter vector is found by solving a least-squares problem, resulting in a stable algorithm. We refer interested readers to [20] for more details on its convergence analysis and performance guarantee.

IV. Optimal Tap Setting Problem as An MDP

In this section, we formulate the optimal tap setting problem as an MDP as follows:

1) State space: Define the squared voltage magnitudes at all buses but bus 0 and the tap ratios as the state, i.e., $\boldsymbol{s} = (\boldsymbol{t},\boldsymbol{v})$, which has both continuous and discrete variables. Then, the state space is $\mathcal{S}\subseteq\mathcal{T}^{L^t}\times\mathbb{R}^N$. At time instant k, the state is $\boldsymbol{s}_k = (\boldsymbol{t}_k,\boldsymbol{v}_k)$.

2) Action space: The actions are the discrete tap ratio changes of the LTCs, i.e., $\boldsymbol{a} = \Delta\boldsymbol{t}$, and the action space $\mathcal{A}$ is the set of all feasible values of LTC tap ratios, which is a subset of $\Delta\mathcal{T}^{L^t}$. Note the size of the action space increases exponentially with the number of LTCs. At time instant k, the action is $\boldsymbol{a}_k = \Delta\boldsymbol{t}_k$, after applying which the new tap ratios at time instant $k+1$ will become $\boldsymbol{t}_{k+1} = \boldsymbol{t}_k + \Delta\boldsymbol{t}_k$.

3) Reward function: The objective of voltage regulation is to minimize the voltage deviation as measured by the L_2 norm. As such, when the system transitions from state $\boldsymbol{s} = (\boldsymbol{t},\boldsymbol{v})$ into state $\boldsymbol{s}' = (\boldsymbol{t}',\boldsymbol{v}')$ after taking action $\boldsymbol{a} = \Delta\boldsymbol{t} := \boldsymbol{t}' - \boldsymbol{t}$, the reward is computed by the following function:

$$\mathcal{R}(\boldsymbol{s},\boldsymbol{a},\boldsymbol{s}') = -\frac{1}{N}\|\boldsymbol{v}' - \boldsymbol{v}^\star\|. \tag{14}$$

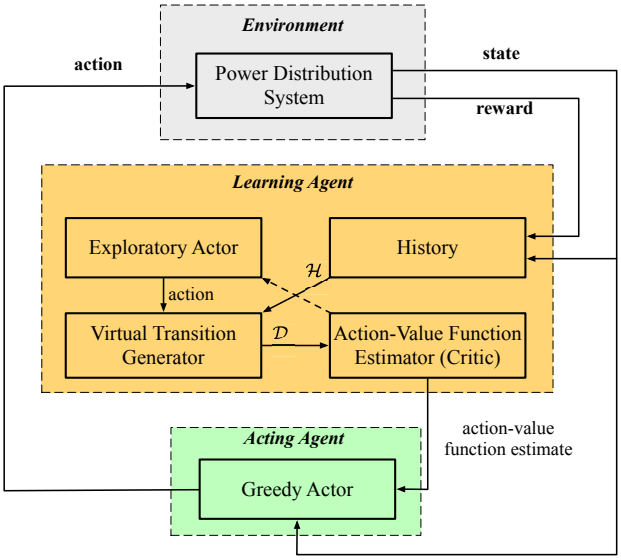

Fig. 2. The batch RL based framework for optimal tap setting. (Dotted line indicates the critic is optional for the exploratory actor.)

4) Transition model: To derive the transition model $\mathcal{P}$, note that it follows from (4) that

$$\boldsymbol{v}' = (\boldsymbol{M}(\boldsymbol{t}')^\top)^{-1}(\boldsymbol{\xi} + \boldsymbol{M}(\boldsymbol{t})^\top \boldsymbol{v} + \boldsymbol{m}(\boldsymbol{t})v_0 - \boldsymbol{m}(\boldsymbol{t}')v_0), \tag{15}$$

where $\boldsymbol{\xi} = 2\mathrm{diag}(\boldsymbol{r})\boldsymbol{M}^{-1}(\boldsymbol{p}' - \boldsymbol{p}) + 2\mathrm{diag}(\boldsymbol{x})\boldsymbol{M}^{-1}(\boldsymbol{q}' - \boldsymbol{q})$, and $\boldsymbol{p}'$ and $\boldsymbol{q}'$ are active and reactive power injections that result into $\boldsymbol{v}'$, respectively. Then, the transition model $\mathcal{P}(\boldsymbol{s}'|\boldsymbol{s}, \boldsymbol{a})$ can be derived from the probability density function (pdf) of $(\boldsymbol{v}'|\boldsymbol{v}, \boldsymbol{t}, \Delta\boldsymbol{t})$, which can be further computed from the pdf of $(\boldsymbol{\xi}|\boldsymbol{v}, \boldsymbol{t}, \Delta\boldsymbol{t})$. However, under Assumptions **A1** and **A2**, the line parameters as well as the probability distributions of active and reactive power injections are unknown; thus, the transition model is not known a priori. Therefore, we need to resort to RL algorithms that do not require an explicit transition model to solve the MDP.

V. Optimal Tap Setting Algorithm

In this section, we propose an optimal tap setting algorithm, which consists of a transition generating algorithm that can generate samples of transitions, and an LSPI-based sequential learning algorithm to solve the MDP. Implementation details such as the feature selection, as well as potential extensions are also discussed.

A. Overview

The overall structure of the optimal tap setting framework is illustrated in Fig. 2. The framework consists of an environment that is the power distribution system, a learning agent that learns the action-value function from a set of transition samples, and an acting agent that determines the optimal action from the action-value function. Define the history to be the sequence of states, actions, and rewards, and denote it by $\mathcal{H}$, i.e., $\mathcal{H} = \{\boldsymbol{s}_0, \boldsymbol{a}_0, r_0, \boldsymbol{s}_1, \boldsymbol{a}_1, r_1, \cdots\}$. Specifically, the learning agent will use the elements in the set $\mathcal{H}$ together with a virtual transition generator to generate a set of transition samples $\mathcal{D}$ according to some exploratory behavior defined in the exploratory actor. The set of transition samples in $\mathcal{D}$ is then used by the action-value function estimator—also referred to as the critic—to fit an approximate action-value function using the LSPI algorithm described earlier.[2] The acting agent, which has a copy of the up-to-date approximate action-value function from the learning agent, finds a greedy action for the current state and instructs the LTCs to follow it.

Note that the learning of the action-value function can be done offline by the learning agent, which is capable of exploring various system conditions through the virtual transition generator based on the history $\mathcal{H}$, yet without directly interacting with the power distribution system. This avoids jeopardizing system operational reliability, which is a major concern when applying RL algorithms to power system applications [17].

B. Virtual Transition Generator

The LSPI algorithm (as well as all other RL algorithms) requires adequate transition samples that spread over the state and action spaces $\mathcal{S} \times \mathcal{A}$. However, this is challenging in power systems since the system operational reliability might be jeopardized when exploring randomly. One way to work around this issue is to use simulation models, rather than the physical system, to generate virtual transitions. To this end, we develop a data-driven virtual transition generator that simulates transitions without any knowledge of the active and reactive power injections (neither measurements nor probability distributions) or the line parameters.

The fundamental idea is the following. For an actual transition sample $(\boldsymbol{s}, \boldsymbol{a}^\dagger, r^\dagger, \boldsymbol{s}^\dagger = (\boldsymbol{t}^\dagger, \boldsymbol{v}^\dagger))$ that is obtained from $\mathcal{H}$, the virtual transition generator generates a new transition sample $(\boldsymbol{s}, \boldsymbol{a}^\ddagger, r^\ddagger, \boldsymbol{s}^\ddagger = (\boldsymbol{t}^\ddagger, \boldsymbol{v}^\ddagger))$, where $\boldsymbol{a}^\ddagger$ is determined from $\boldsymbol{s}$ according to some exploration policy (to be defined later) that aims to explore the state and action spaces. Replacing $\boldsymbol{a}^\dagger$ in the first transition sample with $\boldsymbol{a}^\ddagger$, the voltage magnitudes will change accordingly. Assume the same transition of the power injections in these two samples, then the RHS of (4) does not change. Thus, $\boldsymbol{v}^\ddagger$ can be readily computed from $\boldsymbol{v}^\dagger$ by solving the following set of linear equations:

$$\boldsymbol{M}(\boldsymbol{t}^\ddagger)^\top \boldsymbol{v}^\ddagger + \boldsymbol{m}(\boldsymbol{t}^\ddagger)v_0 = \boldsymbol{M}(\boldsymbol{t}^\dagger)^\top \boldsymbol{v}^\dagger + \boldsymbol{m}(\boldsymbol{t}^\dagger)v_0. \tag{16}$$

Since the only unknown in (16) is $\boldsymbol{v}^\ddagger \in \mathbb{R}$ and $M(\boldsymbol{t}^\ddagger) \in \mathbb{R}^{N\times N}$ is invertible, we can solve for $\boldsymbol{v}^\ddagger$ as follows:

$$\boldsymbol{v}^\ddagger = (\boldsymbol{M}(\boldsymbol{t}^\ddagger)^\top)^{-1}(\boldsymbol{M}(\boldsymbol{t}^\dagger)^\top \boldsymbol{v}^\dagger + \boldsymbol{m}(\boldsymbol{t}^\dagger)v_0 - \boldsymbol{m}(\boldsymbol{t}^\ddagger)v_0). \tag{17}$$

Many efficient algorithms can be applied to solve (17) since $\boldsymbol{M}(\boldsymbol{t}^\ddagger)$ is sparse. To ease notation, we simply write (17) as

$$\boldsymbol{v}^\ddagger = \varphi(\boldsymbol{v}^\dagger, \boldsymbol{t}^\dagger, \boldsymbol{t}^\ddagger). \tag{18}$$

[2]Note that if the transition characteristics of the power injections can be captured by those in the actual transition samples, the transition samples in $\mathcal{D}$ will be sufficient for learning a good action-value function. The subset of history that is utilized to generate $\mathcal{D}$, and the appropriate size of $\mathcal{D}$, need to be determined through simulation studies following the standard train/validation/test procedure that is widely adopted in developing machine learning algorithms.

Algorithm 1: Virtual Transition Generating

Input:
$\mathcal{H}$: history
D: number of transition samples
$\boldsymbol{v}^\star$: reference vector of squared voltage magnitudes
exploration policy
Output:
$\mathcal{D}$: transition sample set
Initialize $\mathcal{D} \leftarrow \varnothing$
for $d = 1, \cdots, D$ **do**
  Choose a transition sample $(\boldsymbol{s}, \boldsymbol{a}^\dagger, r^\dagger, \boldsymbol{s}^\dagger = (\boldsymbol{t}^\dagger, \boldsymbol{v}^\dagger))$ from $\mathcal{H}$
  Select $\boldsymbol{a}^\ddagger$ according to exploration policy and set $\boldsymbol{t}^\ddagger = \boldsymbol{t}^\dagger + \boldsymbol{a}^\ddagger$
  Estimate $\boldsymbol{v}^\ddagger$ following $\boldsymbol{a}^\ddagger$ as $\boldsymbol{v}^\ddagger = \varphi(\boldsymbol{v}^\dagger, \boldsymbol{t}^\dagger, \boldsymbol{t}^\ddagger)$
  Compute the reward by $r^\ddagger = -\frac{1}{N}\|\boldsymbol{v}^\ddagger - \boldsymbol{v}^\star\|$
  Add $(\boldsymbol{s}, \boldsymbol{a}^\ddagger, r^\ddagger, \boldsymbol{s}^\ddagger = (\boldsymbol{t}^\ddagger, \boldsymbol{v}^\ddagger))$ to $\mathcal{D}$
end

This property allows us to estimate the new values of voltage magnitudes when the tap positions change without knowing the exact values of power injections and line parameters. The virtual transition generating procedure is summarized in Algorithm 1.

The virtual transition generator essentially plays the role of a power distribution system simulator that can generate transition samples by enhancing available actual transition samples. Therefore, it can be replaced with a complete system simulation model whenever available. The virtual transition generator fully explores the structural characteristics of the LinDistFlow model for balanced radial power distribution systems, and is computationally efficient since it only requires solving a set of sparse linear equations. Yet, measurements of voltage magnitudes at all buses are required in order to solve (17).

C. LSPI-based Sequential Action-Value Function Learning

We next present a learning algorithm for $\hat{Q}(\boldsymbol{s}, \boldsymbol{a})$ based on the LSPI algorithm. While the LSPI is very efficient when the action space is small, it becomes computationally intractable when the action space is large, since the number of unknown parameters in the approximate action-value function is typically proportional to $|\mathcal{A}|$, where $|\cdot|$ denotes the cardinality of a set, which increases exponentially with the number of LTCs. To overcome the "curse of dimensionality" that results from the size of the action space, we propose an LSPI-based sequential learning algorithm to learn the action-value function.

The key idea is the following. Instead of learning an approximate optimal action-value function for the action vector $\boldsymbol{a}$, we learn a separate approximate action-value function for each component of $\boldsymbol{a}$. To be more specific, for each LTC $l \in \mathcal{L}^t$, we learn an approximate optimal action-value function $\hat{Q}^{(l)}(\boldsymbol{s}, a^{(l)}) = \boldsymbol{\phi}^{(l)}(\boldsymbol{s}, a^{(l)})^\top \boldsymbol{w}^{(l)}$, where $a^{(l)}$ is the l^{th} component of $\boldsymbol{a}$, $\boldsymbol{\phi}^{(l)}(\cdot, \cdot)$ is a feature mapping from $\mathcal{S} \times \Delta\mathcal{T}$ to $\mathbb{R}^f$. During the learning process of $\boldsymbol{w}^{(l)}$, the rest of the LTCs are assumed to behave greedily according to their own approximate optimal action-value function. To achieve this, we design the following exploration policy to generate the virtual transition samples $\mathcal{D}$ used when learning $\boldsymbol{w}^{(l)}$ for LTC l. In the exploration step in Algorithm 1, the tap ratio change of LTC l is selected uniformly in $\Delta\mathcal{T}$ (uniform exploration), while those of others are selected greedily with respect to the up-to-date $\hat{Q}^{(l)}(\cdot, \cdot)$ (greedy exploration).[3]

Algorithm 2: LSPI-based Action-Value Function Learning for Single LTC

Input:
l: index of LTC
$\mathcal{D}$: transition sample set
$\boldsymbol{\phi}(\cdot, \cdot)$: basis function
γ: discount factor
ε: convergence threshold
δ: pre-condition number
$\boldsymbol{w}_0^{(l)}$: initial parameter vector of approximate optimal action-value function for LTC l
Output:
$\boldsymbol{w}^{(l)}$: updated parameter vector of approximate optimal action-value function for LTC l
Initialize $\boldsymbol{w}_{-1}^{(l)} = \mathbf{0}_f$ and $i = 0$
while $\|\boldsymbol{w}_i^{(l)} - \boldsymbol{w}_{i-1}^{(l)}\| > \varepsilon$ *or* $i = 0$ **do**
  Initialize $\boldsymbol{B}_0 = c\boldsymbol{I}_{f\times f}$ and $\boldsymbol{b}_0 = \mathbf{0}_f$, set $j = 1$
  for $(\boldsymbol{s}, \boldsymbol{a}, r, \boldsymbol{s}') \in \mathcal{D}$ **do**
    $a^{(l)'} = \arg\max_{a \in \Delta\mathcal{T}} \boldsymbol{\phi}(\boldsymbol{s}', a)^\top \boldsymbol{w}_i^{(l)}$
    $\boldsymbol{B}_j = \boldsymbol{B}_{j-1} + \boldsymbol{\phi}(\boldsymbol{s}, a^{(l)})(\boldsymbol{\phi}(\boldsymbol{s}, a^{(l)}) - \gamma\boldsymbol{\phi}(\boldsymbol{s}', a^{(l)'}))^\top$
    $\boldsymbol{b}_j = \boldsymbol{b}_{j-1} + \boldsymbol{\phi}(\boldsymbol{s}, a^{(l)})r$
    Increase j by 1
  end
  $\boldsymbol{w}_{i+1}^{(l)} = \boldsymbol{B}_{|\mathcal{D}|}^{-1}\boldsymbol{b}_{|\mathcal{D}|}$, increase i by 1
end
Set $\boldsymbol{w}^{(l)} = \boldsymbol{w}_i^{(l)}$

Then, the LSPI algorithm detailed in Algorithm 2, where c is a small positive pre-condition number and $\boldsymbol{w}_0^{(l)}$ is the initial value for the parameter vector, is applied to learn $\boldsymbol{w}^{(l)}$ for LTC l. This procedure is repeated in a round-robin fashion for all LTCs for J rounds, in each of which $\boldsymbol{w}_0^{(l)}$ is set to the up-to-date $\boldsymbol{w}^{(l)}$ learned in the previous round or initialized to ones if it is in the first round. The value of J is set to 1 if there is only one LTC and is increased slightly when there are more LTCs. Note that a new set of transitions $\mathcal{D}$ is generated when learning $\boldsymbol{w}^{(l)}$ for different LTCs at each round. The complete procedure of the sequential learning algorithm that learns the parameter vectors for all LTCs is presented in Algorithm 3. Using this sequential learning algorithm, the total number of unknowns is then proportional to $L^t|\Delta\mathcal{T}|$, which is far fewer compared to $|\Delta\mathcal{T}^{L^t}|$ as in the case where the approximate

[3]When following either exploration policy, if the selected tap ratio change falls outside of the set of feasible values, it is replaced by the nearest feasible value so as to ensure that the new tap ratio is within $[0.9, 1.1]$.

Algorithm 3: Sequential Action-Value Function Learning for All LTCs

Input:
J: number of learning rounds
$\boldsymbol{w}^{(l)}$: previous parameter vector of approximate optimal action-value function for LTC $l \in \mathcal{L}^t$
Output:
$\boldsymbol{w}^{(l)}$: updated parameter vector of approximate optimal action-value function for LTC $l \in \mathcal{L}^t$
for $j = 1, \cdots, J$ **do**
  for $l = 1, \cdots, L^t$ **do**
    Run Algo. 1 to generate $\mathcal{D}$ using uniform exploration for LTC l and greedy exploration for other LTCs
    Update $\boldsymbol{w}^{(l)}$ by running Algo. 2 with $\boldsymbol{w}_0^{(l)}$ set to the current $\boldsymbol{w}^{(l)}$
  end
end

optimal action-value function for the entire action vector, $\boldsymbol{a}$, is learned.

A critical step in implementing the LSPI algorithm is constructing features from the state-action pair $(\boldsymbol{s}, a^{(l)})$ for LTC l; we use radial basis function (RBFs) to this end. The feature vector for a state-action pair $(\boldsymbol{s}, a^{(l)})$, i.e., $\boldsymbol{\phi}^{(l)}(\boldsymbol{s}, a^{(l)})$, is a vector in $\mathbb{R}^f$, where $f = (\kappa + 1) \times |\Delta\mathcal{T}|$ and κ is a positive integer. The feature vector $\boldsymbol{\phi}^{(l)}(\boldsymbol{s}, a^{(l)})$ has $|\Delta\mathcal{T}|$ segments, each one of length $\kappa + 1$ corresponding to a tap change in $\Delta\mathcal{T}$, i.e, $\boldsymbol{\phi}^{(l)}(\boldsymbol{s}, a^{(l)}) = [\boldsymbol{\psi}_1^\top, \cdots, \boldsymbol{\psi}_{|\Delta\mathcal{T}|}^\top]^\top$, where $\boldsymbol{\psi}_i \in \mathbb{R}^{\kappa+1}, i = 1, \cdots, |\Delta\mathcal{T}|$. Specifically, for $\boldsymbol{s} = (\boldsymbol{t}, \boldsymbol{v})$ and $a^{(l)}$ being the i^{th} tap change in $\Delta\mathcal{T}$, $\boldsymbol{\psi}_j = \mathbf{0}_{\kappa+1}$ for $j \neq i$, and $\boldsymbol{\psi}_i = [1, e^{-\frac{\|\tilde{\boldsymbol{v}} - \bar{\boldsymbol{v}}_1\|}{\sigma^2}}, \cdots, e^{-\frac{\|\tilde{\boldsymbol{v}} - \bar{\boldsymbol{v}}_\kappa\|}{\sigma^2}}]^\top$, where $\sigma > 0$, $\tilde{\boldsymbol{v}} = \varphi(\boldsymbol{v}, \boldsymbol{t}, \tilde{\boldsymbol{t}})$ with $\tilde{\boldsymbol{t}}$ being obtained by replacing the l^{th} entry in $\boldsymbol{t}$ with 1, and $\bar{\boldsymbol{v}}_i$, $i = 1, \cdots, \kappa$ are pre-specified constant vectors in $\mathbb{R}^N$ referred to as the RBF centers. The action $a^{(l)}$ only determines which segment will be non-zero. Thus, $\tilde{\boldsymbol{v}}$ is indeed the squared voltage magnitudes under the same power injections if the tap of LTC l is at position 0. Each RBF computes the distance between $\boldsymbol{v}'$ and some pre-specified squared voltage magnitudes.

The LSPI-based sequential action-value function learning algorithm is not restricted to balanced radial power distribution systems. In fact, it can be readily applied to any power distribution system, possibly unbalanced and with mesh topology, as long as adequate transition samples can be obtained. When applied to three-phase unbalanced distribution systems, the dimension of the state and the action will be tripled as one needs to take into account the three phases at each bus. The same methodology explained in this section, which sequentially learns a separate approximate action-value function for each LTC, can be directly applied to solve the LTC tap setting problem in unbalanced distribution systems. Yet, it is also possible to learn an approximate action-value function for all three LTCs at the same bus, thus reducing the number of action-value functions to be learned. In the latter case, where the "reduced" action is a three-dimensional vector, it is necessary to take advantage of our prior knowledge on the operations of LTCs at the same bus so as to effectively narrow the action space. Since the focus of this paper is on three-phase balanced distribution systems, we leave the extension to unbalanced distribution systems as future work.

Algorithm 4: Tap Setting

Input:
ϵ: minimum action-value change
$\boldsymbol{s}_k$: state at time instant k
Output:
$\boldsymbol{a}_k$: action at time instant k
for $l = 1, \cdots, L^t$ **do**
  if $\max_{a \in \Delta\mathcal{T}} \boldsymbol{\phi}(\boldsymbol{s}_k, a)^\top \boldsymbol{w}^{(l)} - \boldsymbol{\phi}(\boldsymbol{s}_k, a_{k-1}^{(l)})^\top \boldsymbol{w}^{(l)} > \epsilon$ **then**
    Set $a_k^{(l)} = \arg\max_{a \in \Delta\mathcal{T}} \boldsymbol{\phi}(\boldsymbol{s}_k, a)^\top \boldsymbol{w}^{(l)}$
  else
    Set $a_k^{(l)} = a_{k-1}^{(l)}$
  end
end

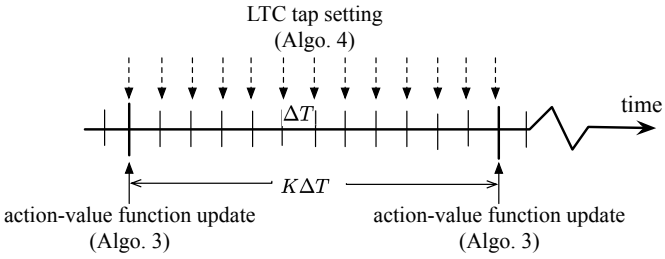

Fig. 3. LTC tap setting procedure.

D. Tap Setting Algorithm

Once the approximate optimal action-value functions have been learned, they can be utilized to determine the tap ratio changes from the state at each time instant. The tap setting procedure illustrated in Fig. 3 is explained as follows. At time instant k, a new state $\boldsymbol{s}_k$ as well as the reward following the action $\boldsymbol{a}_{k-1}$, r_{k-1}, is observed. Let ΔT denote the time elapsed between two time instants. Every K time instants, i.e., every $K\Delta T$ units of time, the parameter vectors $\boldsymbol{w}^{(l)}, l \in \mathcal{L}^t$ are updated by the learning agent by executing the LSPI-based sequential learning algorithm detailed in Algorithm 3. The acting agent then finds a greedy action for the current state $\boldsymbol{s}_k$ and sends it to the LTCs. Note that in order to reduce the wear and tear on the LTCs, the greedy action for the current state $\boldsymbol{s}_k$ is chosen only if the difference between the action-value resulting from the greedy action, i.e., $\max_{a \in \Delta\mathcal{T}} \boldsymbol{\phi}(\boldsymbol{s}_k, a)^\top \boldsymbol{w}^{(l)}$, and that resulting from the previous action, i.e., $\boldsymbol{\phi}(\boldsymbol{s}_k, a_{k-1}^{(l)})^\top \boldsymbol{w}^{(l)}$, is larger than a threshold ϵ. Otherwise, the tap positions do not change. The tap setting algorithm is detailed in Algorithm 4.

VI. Numerical Simulation

In this section, we apply the proposed algorithm to the IEEE 13-bus and 123-bus test feeders from [24].

A. Simulation Setup

All simulations are run on a MacBook Pro with a processor of 2.6 GHz Intel Core i7 and 16 GB memory. The power injections for these two test feeders are constructed based on historical hourly active power load data from a residential building in San Diego over one year [25]. Specifically, the historical hourly active power load data are first scaled up so that the maximum system total active power load over that year for the IEEE 13-bus and 123-bus distribution test feeders are 6.15 MW and 12.3 MW, respectively. These numbers are chosen so that the resulting voltage magnitudes fall outside of the desired range at some time instants. Then, the time granularity of the scaled system total active power load is increased to 5 minutes through a linear interpolation. Each value in the resulting five-minute load data is further multiplied by a normally distributed variable, the mean and standard deviation (SD) of which are 1 and 0.02, respectively. The active power load profile at each bus is constructed by pseudo-randomly redistributing the system total active power load among all load buses. Each load bus is assumed to have a constant power factor of 0.95. While only load variation is considered in the simulation, the proposed algorithm can be directly applied to the case with renewable-based resources, which can be modeled as negative loads.

We first verify the accuracy of the virtual transition generating algorithm. Specifically, assume the voltage magnitudes are known for some unknown power injections under a known tap ratio of 1. Then, when the tap ratio changes, we compute the true voltage magnitudes under the new tap ratio, denoted by $\boldsymbol{v}$, by solving the full ac power flow problem using Matpower [26], and the estimated voltage magnitudes under new tap ratio, denoted by $\hat{\boldsymbol{v}}$, via (18). Simulation results indicate that the voltage approximation error, defined as the maximum absolute difference between the true and the estimated voltage magnitude, i.e., $\|\boldsymbol{v} - \hat{\boldsymbol{v}}\|_\infty$, is smaller than 0.001 p.u., which is accurate enough for the application of voltage regulation addressed in this paper. The voltage approximation error for the IEEE 13-bus test feeder is presented in Fig. 4. The result for the IEEE 123-bus test feeder is similar and therefore is not presented here.

B. Case Study on the IEEE 13-bus Test Feeder

Assume $\boldsymbol{v}^\star = \mathbf{1}_N$, where $\mathbf{1}_N$ is an all-ones vector in $\mathbb{R}^N$ In the simulation, 21 RBF centers are used, i.e., $\kappa = 21$. Specifically, $\bar{\boldsymbol{v}}_i = (0.895 + 0.005i)^2 \times \mathbf{1}_N$, $i = 1, \cdots, 21$. The duration between two time instants is $\Delta T = 5$ min. The policy is updated every 2 hours, i.e., $K = 24$. In each update, actual transition samples are chosen from the history over the same time interval in the previous 5 days, which are part of $\mathcal{H}$, and new actions are chosen according to the exploration policy described in Section V-C. A total number of $D = 6000$ virtual transitions are generated using Algorithm 1. Since this test feeder only has one LTC, there is no need to sequentially update the approximate action-value function, so we set $J = 1$. Other parameters are chosen as follows: $\gamma = 0.9$, $\varepsilon = 1 \times 10^{-5}$, $\epsilon = 1 \times 10^{-4}$, $c = 0.1$, and $\sigma = 1$.

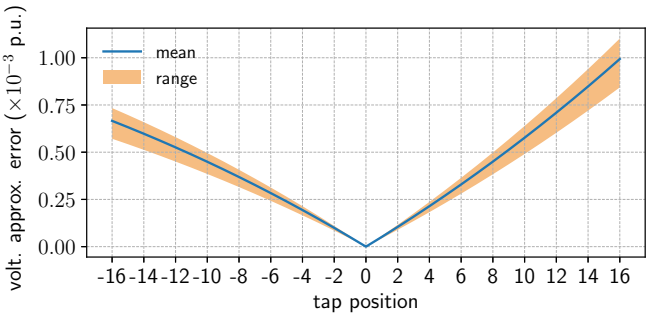

Fig. 4. Voltage approximation error. (Position 0 corresponds to tap ratio 1.)

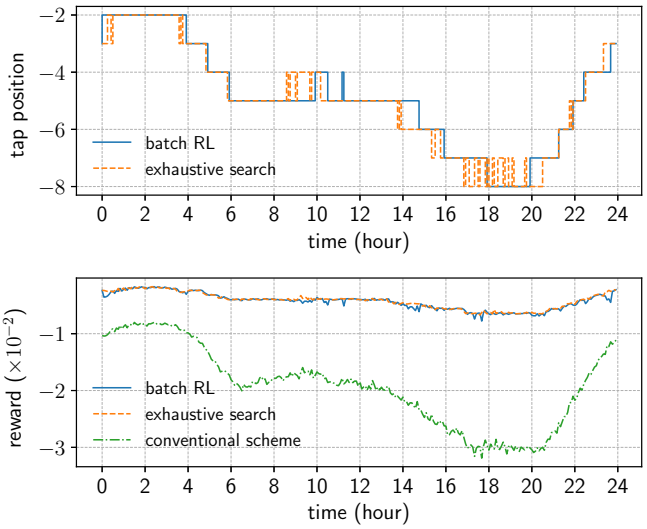

Fig. 5. LTC tap positions and rewards in IEEE 13-bus test feeder.

Assuming complete and perfect knowledge of the system parameters as well as active and reactive power injections for all time instants, we can find the optimal tap position that results in the highest reward by exhaustively searching the action space, i.e., all feasible tap ratios, at each time instant. It is important to point out that, in practice, the exhaustive search approach is infeasible since we do not have the necessary information, and not practical due to the high computational burden. Results obtained by the exhaustive search approach and the conventional tap setting scheme (see, e.g., [1]), in which the taps are adjusted only when the voltage magnitudes exceed a desired range, e.g., $[0.9, 1.1]$ p.u., are used to benchmark the proposed algorithm.

1) Basic Results: Figure 5 shows the tap positions (top panel) and the rewards (bottom panel) under different approaches for the base case. The rewards resulted from these two approaches are very close. The daily mean rewards, i.e., $\rho = \frac{1}{288}\sum_{k=1}^{288} r_k$, where r_k is the reward at time instant k as defined in (14), obtained by the batch RL approach

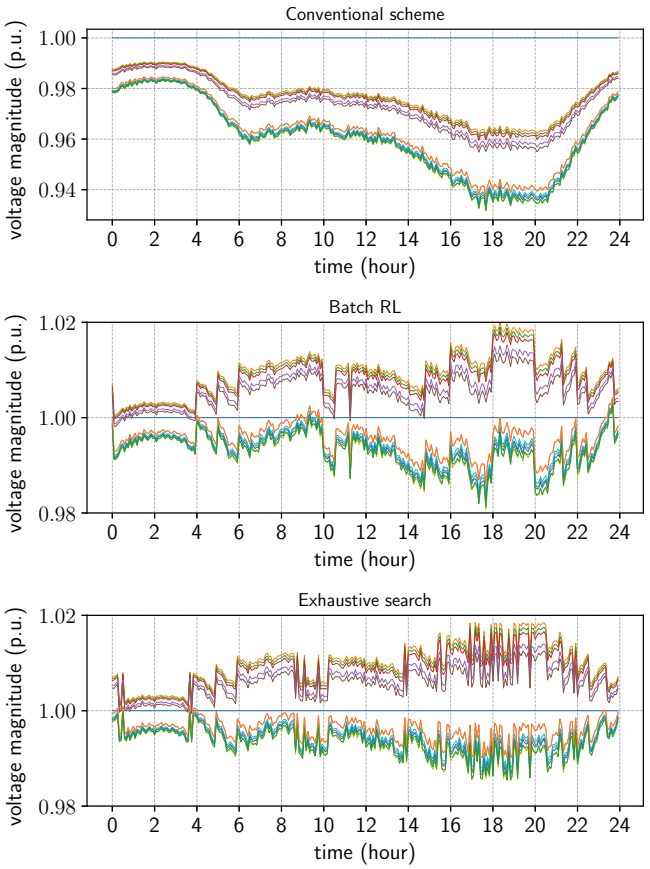

Fig. 6. Voltage magnitude profiles in IEEE 13-bus test feeder.

TABLE I
REWARDS UNDER VARIOUS POLICY UPDATE PERIODS.

update period (hour)	2	4	6	8	24
reward mean ($\times 10^{-3}$)	-4.279	-4.290	-4.390	-4.347	-4.501
reward SD ($\times 10^{-3}$)	1.436	1.423	1.480	1.529	1.601

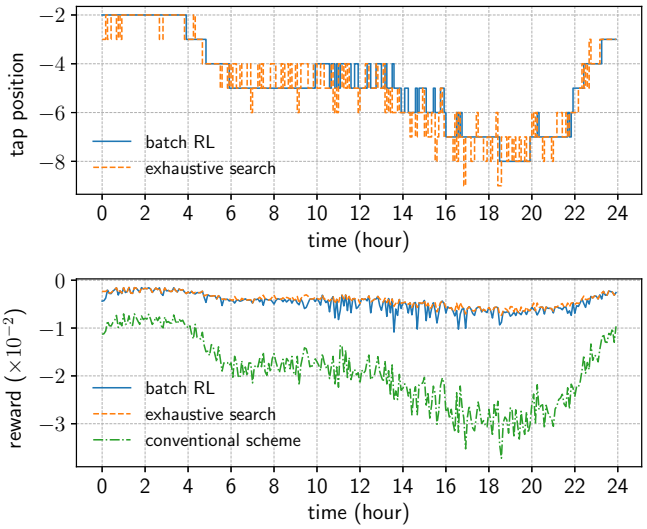

Fig. 7. LTC tap positions and rewards in IEEE 13-bus test feeder with more volatile loads.

and the exhaustive search approach are $\rho = -4.279 \times 10^{-3}$ and $\rho = -4.156 \times 10^{-3}$, respectively, while that under the conventional scheme is $\rho = -19.26 \times 10^{-3}$. In addition, the reward SDs over one day, i.e, $\varsigma = \sqrt{\frac{1}{288}\sum_{k=1}^{288}(r_k - \rho)^2}$, obtained by the batch RL approach, the exhaustive search approach, and the conventional scheme are $\varsigma = 1.436 \times 10^{-3}$, $\varsigma = 1.399 \times 10^{-3}$, and $\varsigma = 7.042 \times 10^{-3}$, respectively. The tap positions under the batch RL approach and the exhaustive search approach are aligned during most of the time during the day. Note that the tap position under the conventional scheme remains at 0 since the voltage magnitudes are within $[0.9, 1.1]$ p.u., and is not plotted. Figure 6 shows the voltage magnitude profiles under the different tap setting algorithms. The voltage magnitude profiles under the proposed batch RL algorithm (see Fig. 6, center panel) are quite similar to those obtained via the exhaustive search approach (see Fig. 6, bottom panel); both result in a higher daily mean reward than that resulted from the conventional scheme (see Fig. 6, top panel).

Algorithm 3 typically converges in about 30 seconds, and the batch RL approach is faster than the exhaustive search approach by several orders of magnitude. The time it takes to generate the virtual transition samples using Algorithm 1 is negligible. Yet, a substantial amount of computation time—much more than that required by Algorithm 2—is needed if the same number of transition samples are to be generated by solving the full ac power flow problem using a complete system model. This shows the proposed algorithm is indeed computationally efficient.

2) Impacts of Policy Update Periods: While the policy in the above simulation is updated using data over 2 hours, we also study cases when the policy is update using data over a longer time period, i.e., when K is larger than 24. Table I shows the mean and SD of rewards over one day when the policy is updated using data over different periods, when $|\mathcal{D}|$ is 50 times the number of actual transitions during the same time period over the previous 5 days. As can be seen, learning a policy using data over a shorter period reduces the required amount of data, while achieving an equally good (or even better) result.

3) Impacts of Load Variability: To see the impacts of load variability on the performance of the proposed batch RL algorithm, we run simulation for a case with more volatile loads. To obtain more volatile loads, we multiply each value in the original five-minute load data by a normally distributed variable, the mean and SD of which are 1 and 0.1, respectively. The rest of the simulation procedure is the same as the base case.

Figure 7 shows the tap positions (top panel) and the rewards (bottom panel) under different approaches. The daily mean rewards obtained by the batch RL approach, the exhaustive search approach, and the conventional scheme are $\rho = -4.648 \times 10^{-3}$, $\rho = -4.121 \times 10^{-3}$, and $\rho = -19.12 \times 10^{-3}$, respectively. In addition, the SDs of rewards over one day obtained by the batch RL approach, the exhaustive search approach, and the conventional scheme are $\varsigma = 1.825 \times 10^{-3}$,

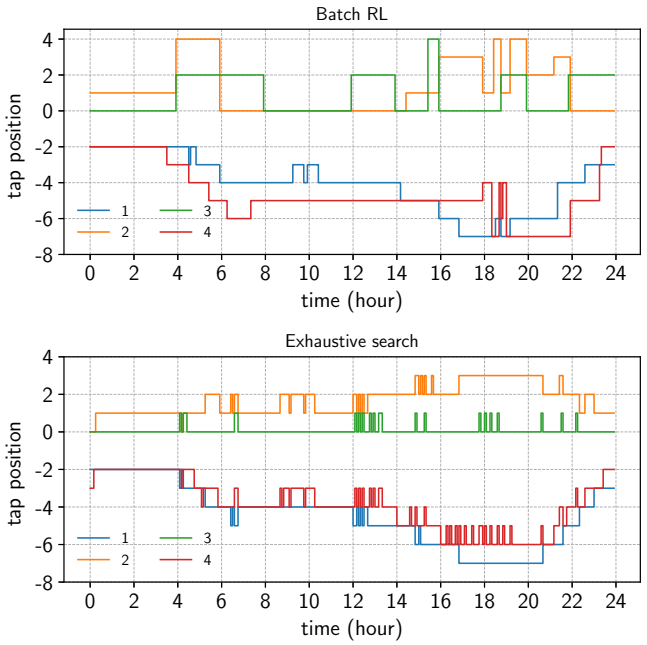

Fig. 8. LTC tap positions in IEEE 123-bus test feeder.

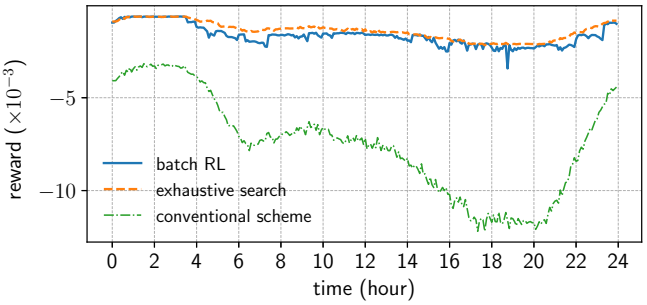

Fig. 9. Rewards in IEEE 123-bus test feeder.

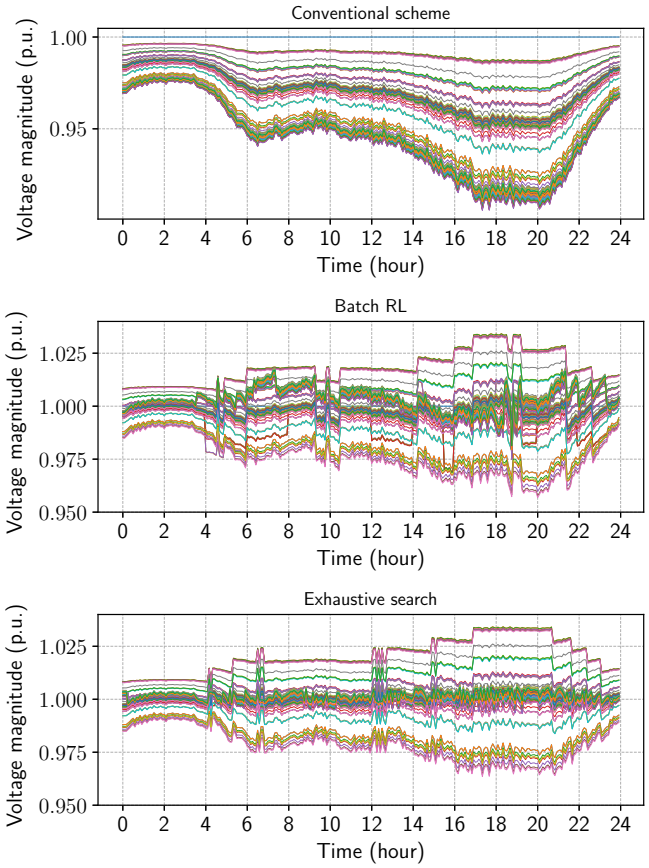

Fig. 10. Voltage magnitude profiles in IEEE 123-bus distribution test feeder.

$\varsigma = 1.389 \times 10^{-3}$, and $\varsigma = 7.138 \times 10^{-3}$, respectively. Compared to the base case with less volatile loads, the mean reward obtained by the batch RL approach decreased and the SD of rewards increased slightly, yet they are still close to those obtained by the exhaustive search approach, which demonstrates the robustness of the proposed algorithm against load variability.

C. Case Study on the IEEE 123-bus Test Feeder

We next test the proposed batch RL algorithm on the IEEE 123-bus test feeder. In the results for the IEEE 13-bus test feeder reported earlier, while the LTC has 33 tap positions, only a small portion of them is actually used. This motivates us to further reduce the action space by narrowing the action space to a smaller range. Specifically, we can estimate the voltage magnitudes under various power injections and LTC tap positions using (18). After ruling out tap positions under which the voltage magnitudes will exceed the desired range, we eventually allow 9 positions, from -8 to 0, for two LTCs, and 5 positions, from 0 to 5, for the other two LTCs. Here, $\kappa = 11$ RBF centers are used. Specifically, $\bar{\boldsymbol{v}}_i = (0.94+0.01i)^2 \times \mathbf{1}_N$ for all LTCs except for the one near the substation, for which $\bar{\boldsymbol{v}}_i = (0.89+0.01i)^2 \times \mathbf{1}_N$, $i = 1, 2, \cdots, 11$. A total number of $D = 3600$ virtual transitions are generated in a similar manner as in the IEEE 13-bus test feeder case. The number of rounds in the LSPI-based sequential learning algorithm is set to $J = 3$. Note that the appropriate value of J is affected by the number of LTCs rather than the number of buses. Since typical power distribution systems have 1 to 4 LTCs (see [24]), setting J to 3 will be generally adequate. Other parameters are the same as in the IEEE 13-bus test feeder case.

Figures 8 and 9 show the tap positions and rewards, respectively, under the batch RL approach and the exhaustive search. The daily mean rewards obtained by the batch RL approach and the exhaustive search approach are $\rho = -1.646 \times 10^{-3}$ and $\rho = -1.386 \times 10^{-3}$, respectively, while that under the conventional scheme is $\rho = -7.513 \times 10^{-3}$. In addition, the reward SDs over one day obtained by the batch RL approach, the exhaustive search approach, and the conventional scheme are $\varsigma = 5.424 \times 10^{-4}$, $\varsigma = 4.713 \times 10^{-4}$, and $\varsigma = 2.706 \times 10^{-3}$, respectively. While the tap positions differ, the rewards resulting from these two approaches are very close. Note that the tap changes are smoother in the proposed algorithm since we have enforced a minimum action-value function change requirement to trigger a tap change action. The voltage magnitude profiles are shown in Fig. 10.

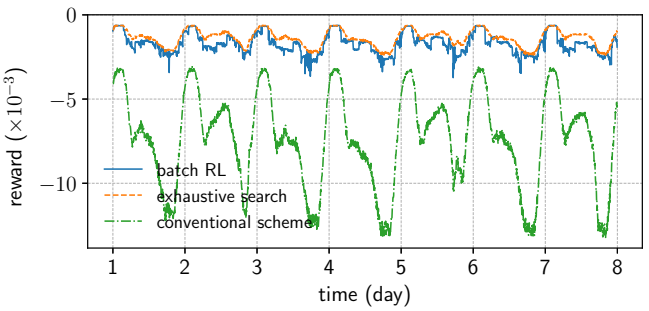

Fig. 11. Rewards in IEEE 123-bus test feeder over 7 days.

In this case, each round in Algorithm 3 typically takes less than 10 seconds, and the total time it takes for $J = 3$ rounds is less than 30 seconds. Despite the fact that the number of LTCs in this test feeder is more than that in the IEEE 13-bus test feeder, the run times of Algorithm 3 in both test feeders are close. This is due to a reduction in the number of transition samples used during training and in the size of the action space, as well as the fact the generating virtual transitions using Algorithm 1 is computationally efficient.

To see the performance of the proposed algorithm over a longer time period, we also run it for one week under the same simulation setup. Figure 11 shows the rewards over one week. As can be seen from Fig. 11, the voltage regulation performance achieved by the proposed algorithm is in general close to that achieved by the exhaustive search approach.

VII. Concluding Remarks

In this paper, we formulated the optimal tap setting problem of LTCs in power distribution systems as an MDP and proposed a batch RL algorithm to solve it. To circumvent the "curse of dimensionality", we proposed an LSPI-based sequential learning algorithm to learn an action-value function for each LTC, based on which the optimal tap positions can be determined directly. To obtain adequate state-action samples, we developed a virtual transition generator that estimates the voltage magnitudes under different tap settings. The proposed algorithm can find from historical data a policy that determines the optimal tap positions that minimize the voltage deviation across the system, based only on voltage magnitude measurements and network topology information, which makes it more desirable for implementation in practice. The optimal policy can be learned offline where most computational burden takes place, and the required computation to find the optimal tap positions is minimal when executed online. Numerical simulation on the IEEE 13- and 123-bus test feeders validated the effectiveness, and the efficiency of the proposed algorithm.

There are several potential directions for future work. The first is to further reduce the information required by the algorithm, specifically, dropping the requirement on topology information. This may be feasible since it is possible to identify the topology of the power distribution system from voltage measurements (see, e.g., [27]).

The second is to develop a distributed version of the LTC tap setting algorithm, which allows determining the optimal tap ratios of each LTC based on a subset of measurements, potentially from the neighboring area of each LTC. Multi-agent RL techniques may be leveraged in such a setting.

The third is to extend the proposed algorithm to general power distribution systems with potentially mesh topology. The LSPI-based sequential learning algorithm can be readily applied to unbalanced power distribution systems and possibly a mesh network. Therefore, the key is to develop a transition sample simulator for general power distribution systems. A complete system simulation model, if available, can be used as a transition sample simulator for training the batch RL algorithm. Alternatively, supervised learning techniques can be leveraged to learn a transition sample simulator from measurements of pertinent variables, including power injections, LTC tap ratios, as well as voltage magnitudes. With the transition sample simulator, the proposed optimal tap setting algorithm can be directly applied, possibly requiring a smaller number of voltage magnitude measurements.

References

[1] P. Kundur, N. J. Balu, and M. G. Lauby, *Power System Stability and Control*. McGraw-hill New York, 1994, vol. 7.

[2] C. R. Sarimuthu, V. K. Ramachandaramurthy, K. Agileswari, and H. Mokhlis, "A review on voltage control methods using on-load tap changer transformers for networks with renewable energy sources," *Renew. Sust. Energ. Rev.*, vol. 62, pp. 1154–1161, 2016.

[3] W.-H. Liu, A. D. Papalexopoulos, and W. F. Tinney, "Discrete shunt controls in a Newton optimal power flow," *IEEE Trans. Power Syst.*, vol. 7, no. 4, pp. 1509–1518, 1992.

[4] M. Salem, L. Talat, and H. Soliman, "Voltage control by tap-changing transformers for a radial distribution network," *IEE P-Gener. Transm. D.*, vol. 144, no. 6, pp. 517–520, 1997.

[5] M. Liu, C. A. Canizares, and W. Huang, "Reactive power and voltage control in distribution systems with limited switching operations," *IEEE Trans. Power Syst.*, vol. 24, no. 2, pp. 889–899, 2009.

[6] B. A. Robbins, H. Zhu, and A. D. Domínguez-García, "Optimal tap setting of voltage regulation transformers in unbalanced distribution systems," *IEEE Trans. Power Syst.*, vol. 31, no. 1, pp. 256–267, 2016.

[7] Z. Yang, A. Bose, H. Zhong, N. Zhang, Q. Xia, and C. Kang, "Optimal reactive power dispatch with accurately modeled discrete control devices: A successive linear approximation approach," *IEEE Trans. Power Syst.*, vol. 32, no. 3, pp. 2435–2444, 2017.

[8] H. Xu, A. Dominguez-Garcia, and P. W. Sauer, "Data-driven coordination of distributed energy resources for active power provision," *IEEE Trans. Power Syst.*, 2019.

[9] D. Ernst, M. Glavic, and L. Wehenkel, "Power systems stability control: reinforcement learning framework," *IEEE Trans. Power Syst.*, vol. 19, no. 1, pp. 427–435, 2004.

[10] J. G. Vlachogiannis and N. D. Hatziargyriou, "Reinforcement learning for reactive power control," *IEEE Trans. Power Syst.*, vol. 19, no. 3, pp. 1317–1325, 2004.

[11] Y. Xu, W. Zhang, W. Liu, and F. Ferrese, "Multiagent-based reinforcement learning for optimal reactive power dispatch," *IEEE Trans. Syst., Man, Cybern., Syst., Part C (Applications and Reviews)*, vol. 42, no. 6, pp. 1742–1751, 2012.

[12] Z. Wen, D. O'Neill, and H. Maei, "Optimal demand response using device-based reinforcement learning," *IEEE Trans. Smart Grid*, vol. 6, no. 5, pp. 2312–2324, 2015.

[13] F. Ruelens, B. J. Claessens, S. Vandael, B. De Schutter, R. Babuška, and R. Belmans, "Residential demand response of thermostatically controlled loads using batch reinforcement learning," *IEEE Trans. Smart Grid*, vol. 8, no. 5, pp. 2149–2159, 2017.

[14] W. Liu, P. Zhuang, H. Liang, J. Peng, and Z. Huang, "Distributed economic dispatch in microgrids based on cooperative reinforcement learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 6, pp. 2192–2203, 2018.

[15] H. Xu, H. Sun, D. Nikovski, S. Kitamura, K. Mori, and H. Hashimoto, "Deep reinforcement learning for joint bidding and pricing of load serving entity," *IEEE Trans. Smart Grid*, 2019.

[16] H. Xu, X. Li, X. Zhang, and J. Zhang, "Arbitrage of energy storage in electricity markets with deep reinforcement learning," *arXiv preprint arXiv:1904.12232*, 2019.
[17] M. Glavic, R. Fonteneau, and D. Ernst, "Reinforcement learning for electric power system decision and control: Past considerations and perspectives," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 6918–6927, 2017.
[18] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. MIT press, 2018.
[19] M. E. Baran and F. F. Wu, "Network reconfiguration in distribution systems for loss reduction and load balancing," *IEEE Trans. Power Del.*, vol. 4, no. 2, pp. 1401–1407, 1989.
[20] M. G. Lagoudakis and R. Parr, "Least-squares policy iteration," *J. Mach. Learn. Res.*, vol. 4, pp. 1107–1149, Dec 2003.
[21] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, no. 3-4, pp. 279–292, 1992.
[22] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.
[23] D. Ernst, P. Geurts, and L. Wehenkel, "Tree-based batch mode reinforcement learning," *J. Mach. Learn. Res.*, vol. 6, no. Apr, pp. 503–556, 2005.
[24] IEEE distribution test feeders. [Online]. Available: http://sites.ieee.org/pes-testfeeders/resources/
[25] Commercial and residential hourly load profiles for all TMY3 locations in the United States. [Online]. Available: https://openei.org/doe-opendata/dataset
[26] R. D. Zimmerman, C. E. Murillo-Sánchez, and R. J. Thomas, "Matpower: Steady-state operations, planning, and analysis tools for power systems research and education," *IEEE Trans. Power Syst.*, vol. 26, no. 1, pp. 12–19, 2010.
[27] D. Deka, S. Backhaus, and M. Chertkov, "Structure learning in power distribution networks," *IEEE Trans. Control Netw. Syst.*, vol. 5, no. 3, pp. 1061–1074, 2018.

Hanchen Xu received the B.Eng. and M.S. degrees in electrical engineering from Tsinghua University, Beijing, China, in 2012 and 2014, respectively, and the M.S. degree in applied mathematics and Ph.D. degree in electrical engineering from the University of Illinois at Urbana-Champaign, Urbana, IL, USA, in 2017 and 2019, respectively. His research interests include control, optimization, reinforcement learning, with applications to power systems and electricity markets.

Alejandro D. Domínguez-García (S'02, M'07) received the degree of electrical engineering from the University of Oviedo (Spain) in 2001 and the Ph.D. degree in electrical engineering and computer science from the Massachusetts Institute of Technology, Cambridge, MA, in 2007.

He is Professor with the Department of Electrical and Computer Engineering (ECE), and Research Professor with the Coordinated Science Laboratory and the Information Trust Institute, all at the University of Illinois at Urbana-Champaign. He is affiliated with the ECE Power and Energy Systems area, and has been a Grainger Associate since August 2011.

His research interests are in the areas of system reliability theory and control, and their applications to electric power systems, power electronics, and embedded electronic systems for safety-critical/fault-tolerant aircraft, aerospace, and automotive applications.

Dr. Domínguez-García received the NSF CAREER Award in 2010, and the Young Engineer Award from the IEEE Power and Energy Society in 2012. In 2014, he was invited by the National Academy of Engineering to attend the US Frontiers of Engineering Symposium, and was selected by the University of Illinois at Urbana-Champaign Provost to receive a Distinguished Promotion Award. In 2015, he received the U of I College of Engineering Dean's Award for Excellence in Research.

He is currently an editor for the IEEE Transactions on Control of Network Systems; he also served as an editor of the IEEE Transactions on Power Systems and IEEE Power Engineering Letters from 2011 to 2017.

Peter W. Sauer (S'73, M'77, SM'82, F'93, LF'12) obtained his Bachelor of Science degree in electrical engineering from the University of Missouri at Rolla in 1969. From 1969 to 1973, he was the electrical engineer on a design assistance team for the Tactical Air Command at Langley Air Force Base, Virginia. He obtained the Master of Science and Ph.D. degrees in Electrical Engineering from Purdue University in 1974 and 1977 respectively. From August 1991 to August 1992 he served as the Program Director for Power Systems in the Electrical and Communication Systems Division of the National Science Foundation in Washington D.C. He is a cofounder of the Power Systems Engineering Research Center (PSERC) and the PowerWorld Corporation. He is a registered Professional Engineer in Virginia and Illinois, a Fellow of the IEEE, and a member of the U.S. National Academy of Engineering. He is currently the Grainger Chair Professor of Electrical Engineering at Illinois. Additional information can be found at: https://ece.illinois.edu/directory/profile/psauer.