Dynamic Distribution Network Reconfiguration Using Reinforcement Learning

Yuanqi Gao, Jie Shi, Wei Wang, and Nanpeng Yu
Department of Electrical and Computer Engineering
University of California, Riverside
Riverside, California 92507
Email: ygao024@ucr.edu, jshi005@ucr.edu, wwang031@ucr.edu, nyu@ece.ucr.edu

***Abstract*—Dynamic distribution network reconfiguration (DNR) algorithms perform hourly dynamic status changes of sectionalizing and tie switches to reduce network line losses, minimize loss of load, or increase hosting capacity for distributed energy resources. Existing algorithms in this field have demonstrated good results when network parameters are assumed to be known. However, in practice inaccurate distribution network parameter estimates are prevalent. This paper solves the minimum loss dynamic DNR problem without the network parameter information. We formulate the DNR problem as a Markov decision process problem and train an off-policy reinforcement learning (RL) algorithm based on historical operation data set. In the online execution phase, the trained RL agent determines the best network configuration at any time step to minimize the expected total operational cost over the planning horizon, which includes the switching costs. To improve the RL algorithm's performance, we propose a novel data augmentation method to create additional synthetic training data based on the existing data set. We validate the proposed framework on a 16-bus distribution test feeder with synthetic data. The learned control policy not only reduces the network loss but also improves the voltage profile.**

***Index Terms*—Dynamic distribution network reconfiguration, reinforcement learning, Q-learning, training data augmentation.**

I. Introduction

By changing the status of remotely controllable switches (RCSs), distribution network reconfiguration (DNR) algorithms improve the distribution system performance. They are particularly useful when a large number of distributed renewable resources (DERs) are installed in the distribution system [1]. Both federal sponsored programs and market forces are facilitating the wide-spread adoption of smart grid technologies such as the advanced metering infrastructure and RCSs [2]. These two technologies are enabling remote data collection and actuation of switches which are critical to the implementation of distribution network reconfiguration.

Network reconfiguration problems can be classified into static DNR [3] and dynamic DNR [4]. The former determines the best network configuration for a given injection pattern; the latter determines an optimal sequence of network configurations over the operational horizon in terms of line resistive loss, loss of load, or hosting capacity for DERs. Common operational constraints that must be modeled in DNR problems consist of voltage magnitude limits and network radiality. Frequent operations of RCSs improve the control objective, they also lead to excessive wear and tear on the equipment. Therefore, the number of switching actions are typically constrained or jointly optimized with the typical control objectives to prevent excessive switching. In this paper, we focus on the problem of line loss minimization through dynamic DNR.

Most of the existing literature on the dynamic DNR problem adopt a physical model-based control approach. They can be divided into three groups based on their solution methods. The first group of literature formulates the dynamic DNR as a mixed-integer or dynamic programming problem. In [5], a polyhedral approximation based mixed-integer linear programming (MILP) formulation is presented. A stochastic MILP formulation is proposed in [6] to address the uncertainties in the electric loads and distributed generations. In some cases, the total number of possible radial configurations is not too large and it is possible to enumerate them. Dynamic programming methods can achieve very effective solutions in those cases [7] [8]. The second group of literature proposes heuristic methods. In [9], the optimal time points for DNR are determined based on a gradual approaching method. A minimum spanning tree method is proposed to solve the DNR problem in [10], where the edge weights are the current magnitudes obtained from the power flow results of the meshed network. In the third group, metaheuristic methods, such as the quantum particle swarm optimization [11] and genetic algorithm [12] are used.

The existing dynamic DNR algorithms rely heavily on accurate knowledge of grid topology and parameters. However, it is difficult for regional electric utilities to maintain accurate network models [13], [14]. The primary and secondary networks' parameter estimates are particularly unreliable [15]. In addition, the computational complexity of the existing physical model-based approaches is typically large.

To cope with unreliable distribution network parameters, we propose a deep reinforcement learning (RL) framework to learn and execute dynamic DNR without the use of distribution network parameter information. One of the major limitations of existing RL algorithms is the low sample efficiency. To address this challenge, we propose an innovative approach to augment past grid operational experiences with synthetic ones. Simulation results on a 16-bus distribution feeder reveal that the proposed deep RL is capable of finding a decent control policy without using the network parameter information.

978-1-5386-8099-5/19/$31.00 ©2019 IEEE

The unique contributions of this paper are listed below.

• Our proposed off-policy RL algorithm is capable of performing dynamic DNR with only network topology information and historical operation data set.

• Our operation experience augmentation technique greatly improves the performance of deep RL algorithms.

The rest of the paper is organized as follows. Section II formulates the distribution network dynamic reconfiguration problem. Section III presents the proposed reinforcement learning algorithm. Section IV shows the simulation results. Section V states the conclusion.

II. Problem Formulation

In this section, we formulate the dynamic DNR as a Markov decision process (MDP) [16]. First, we briefly review some basic concepts of MDPs and introduce our notations. Next, we present the problem formulation.

The dynamic DNR problem can be treated as a sequential decision making process. At each time step, the control algorithm reconfigures the network to produce a new configuration. This intuitive description of the dynamic DNR problem can be converted into precise mathematical statements via MDPs.

An MDP is a tuple ($\mathcal{S}$, $\mathcal{A}$, P, r, γ, T), which consists of a set of states $\mathcal{S}$, a set of actions $\mathcal{A}$, a state transition probability $P(s'|s,a)$ $\forall s', s \in \mathcal{S}$, $a \in \mathcal{A}$, a reward function $r(s,a) : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$, $\forall s \in \mathcal{S}$, $a \in \mathcal{A}$, a discount factor $\gamma \in [0,1]$, and a time horizon T. In an MDP, an agent selects an action $A_t \in \mathcal{A}$ based on the environment's state $S_t \in \mathcal{S}$ at each discrete time step t. After that the agent receives a reward $R_{t+1} = r(S_t, A_t)$ and the environment's state will advance to S_{t+1} according to the state transition probability $P(S_{t+1}|S_t, A_t)$. The process terminates when $t = |T|$ and $S_{|T|}$ is a terminal state. In dynamic DNR problem, the state S_t corresponds to the status of the distribution network, which includes the current topology configuration, loads, and global time. The action A_t can be represented by changing the configuration of the distribution network in S_t. The reward $R_{t+1} = r(S_t, A_t)$ is a numerical measure of how good the reconfiguration action A_t was, in terms of minimizing network loss while keeping the number of switching actions small. The state, action, and reward of the dynamic DNR problem will be clearly defined later in this section.

The goal of the agent is to find a control policy $\pi(a|s)$ in each state s that maximizes the expected discounted return (or state-value function) $V^\pi(s) = E_\pi\left[\sum_{t=0}^{T} \gamma^t R_{t+1} | S_0 = s\right]$, which captures the expected discounted return for starting at state s and following policy π thereafter. For a deterministic policy, π maps each state $s \in \mathcal{S}$ to an action $a \in \mathcal{A}$. For a probabilistic policy, π maps each state to a probability distribution of selecting each action. Another useful quantity is the action-value function $Q^\pi(s,a) = E_\pi\left[\sum_{t=0}^{T} \gamma^t R_{t+1} | S_0 = s, A_0 = a\right]$, which is related to $V^\pi(s)$ by $V^\pi(s) = E_{a\sim\pi(a|s)}\left[Q^\pi(s,a)\right]$. We will discuss the algorithms for solving the MDP problem in Section III. In the rest of this section, we focus on formulating the dynamic DNR problem as an MDP.

We first introduce some notations for the dynamic DNR. Consider a distribution network with n load nodes (PQ buses) and n^0 substations (slack buses). Let v_{it}, p_{it}, and q_{it} be the nodal voltage magnitude, real and reactive power injections of node i at time t. Define vectors p_t and q_t as $p_t = [p_{1t}, p_{2t}, \cdots, p_{nt}]$ and $q_t = [q_{1t}, q_{2t}, \cdots, q_{nt}]$. Let p_t^l be the network's total real line losses at t. We denote a radial configuration of the distribution network at time t by a_t^r. That is, a_t^r represents a rooted spanning forest of the graph associated with the no-shunt distribution network [8]. Each root corresponds to a substation.

Next, we construct the dynamic DNR problem as an MDP as follows. We define the state at time t to be $S_t = [p_t, q_t, a_{t-1}^r, t]$ the action A_t as changing the topology of the network to a_t^r. Therefore, $\mathcal{S}$ consists of the set of all injection patterns together with the set of all possible radial configurations. The latter is also equal to $\mathcal{A}$. The reward function reflects both the network loss and the switching cost and is defined as

$$r(S_t, A_t = a_t^r) = -C^l(p_t, q_t, a_t^r) - C^s(a_{t-1}^r, a_t^r) \quad (1)$$

where C^l is the cost associated with the network loss and C^s is the cost incurred by the change of network configuration. The detailed formulation of C^l and C^s will be shown in Section IV. The expected discounted return $E\left[\sum_{t=0}^{T} \gamma^t r(S_t, A_t)\right]$ with some initial configuration a_{-1}^r for the dynamic DNR problem includes both network losses and switching costs. This completes the construction of the MDP.

Note that the injection patterns p_t and q_t time series might not be strictly Markovian. Nevertheless, we shall still use this definition of S_t because the algorithms that we will be discussing are still applicable even if the Markovian property is slightly violated in practice [16].

During the distribution network reconfiguration process, we need to ensure that the nodal voltages always stay within allowable range. In the MDP framework, physical constraints are typically modeled via a constraint function $V_C^\pi(s) = E_\pi\left[\sum_{t=0}^{T} \gamma^t C_{t+1} | S_0 = s\right]$, where $C_{t+1} = c(S_t, A_t)$ is the amount of constraint violation at time t. We define $c(S_t, A_t)$ to be the sum of absolute value of voltage violations at all metered nodes:

$$c(S_t, A_t) = \sum_{i \in N^v} \left[\max(0, v_{it} - \bar{v}) + \max(0, \underline{v} - v_{it})\right] \quad (2)$$

where N^v is the set of all nodes that have voltage measurement devices; $\bar{v}$ and $\underline{v}$ are the upper and lower bounds for voltage. Now the dynamic DNR can be formulated as a constrained MDP problem:

$$\max_{\pi} \quad V^\pi(s) \quad \text{s.t. } V_C^\pi(s) \le 0 \quad (3)$$

In this paper, we approximately solve (3) by a simplified version of the Lagrangian relaxation. The Lagrangian of (3) is given by:

$$V^\pi(s) - \lambda V_C^\pi(s) = E_\pi\left[\sum_{t=0}^{T} \gamma^t (R_{t+1} - \lambda C_{t+1}) | S_0 = s\right] \quad (4)$$

The RHS of (4) has the same form as $V^{\pi}(s)$, except that the reward R_{t+1} is replaced by $R_{t+1} - \lambda C_{t+1}$. Therefore, for any fixed $\lambda \geq 0$ value, (4) can be maximized by solving the unconstrained MDP, which replaces the original reward function by the following augmented reward function:

$$r(S_t, A_t, \lambda) \doteq r(S_t, A_t) - \lambda c(S_t, A_t) \tag{5}$$

However, to solve (3) exactly, the multiplier $\lambda \geq 0$ needs to be worked out instead of kept fixed. Nevertheless, we shall use a fixed λ and solve the unconstrained MDP with (5) as the reward function in this initial study.

III. Technical Methods

In this section, we review basic concepts of deep reinforcement learning and describe an algorithm to find radial configurations.Then, we present a noval operational experience data generation algorithm. At last, we summarize all technical methods in our proposed RL based dynamic DNR algorithm.

A. Deep Q-Learning

In this subsection, an off-policy RL algorithm will be developed to solve the dynamic DNR problem. An off-policy RL algorithm learns the value of optimal policy independently of the actions took by the agent, whereas an on-policy method works by improving the policy that is used to take actions. In this study, an off-policy RL algorithm is more suitable than an on-policy one for the following two reasons. First, off-policy RL algorithms enable distribution operators to learn from a wealth of historical network reconfiguration operation data. Second, off-policy RL algorithms have much higher sample efficiency than that of the on-policy ones.

One of the most widely used off-policy RL algorithm for MDP problems is the Q-learning, which updates the action-value function iteratively:

$$\begin{aligned} Q(S_t, A_t) \leftarrow & Q(S_t, A_t) + \alpha[R_{t+1} \\ & + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)] \end{aligned} \tag{6}$$

with $Q(S_{t+1}, a) = 0$ if S_{t+1} is a terminal state. This allows the learned action-value function, Q, to converge to the optimal action-value function Q^* provided that all state-action pairs continue to be updated. Once the optimal action-value functions are learned, the optimal control policy $\pi^*(s)$, which maximizes $V^{\pi}(s)$, can be found by:

$$\pi^* : S_t \mapsto \operatorname{argmax}_a \, Q^*(S_t, a) \tag{7}$$

However, it is infeasible to directly apply Q-learning for dynamic DNR problems. This is because even if we discretize the continuous state variables, the dimensionality of the state space still increases exponentially. To deal with high-dimensional state space and continuous state variables, we parameterize an approximate action-value function $Q(S_t, A_t; \theta^Q)$ with a neural network, where θ^Q are the parameters of the neural network.

Nonetheless, this brings its own challenges. Divergence may occur during learning [17]. One cause of divergence is the high correlations between the action values $Q(S_t, A_t)$ and the target values $R_{t+1} + \gamma \max_a Q(S_{t+1}, a)$. To ease this, we adopt a target Q network whose parameters θ^{Q-} are only updated every C steps [18] by $\theta^{Q-} \leftarrow \theta^Q$. The θ^Q update uses the loss function $L(\theta^Q) \doteq E\left[(r + \gamma \max_{a'} Q(s', a'; \theta^{Q-}) - Q(s, a; \theta^Q))^2\right]$, with $Q(s', a', \theta^{Q-}) = 0$ if s' is a terminal state. To further reduce the high correlation, we adopt the replay mechanism [17]. As such, we store the past operational experiences for network reconfiguration $e_t = (S_t, A_t, R_{t+1}, S_{t+1})$ in a 'memory data set' $D_H \doteq \{e_1, ..., e_H\}$, which is sampled during learning. Each sample forms a replay in the learning process.

B. Finding Radial Configurations

When constructing the action domain, all feasible radial configurations need to be enumerated. For single substation distribution networks, this could be done by the tree enumeration algorithm [19, p.464]. For distribution networks with multiple substations, we enhance the algorithm by adding a merge and a split step. Fig.1 provides an example of this enhanced algorithm. First, we merge all the substation nodes (0 and 1) into a single root node X. Then we enumerate spanning trees on the resulting graph. Finally, we split the root node X by identifying the branch-node connectivity on the original graph. This algorithm guarantees that all the rooted spanning forests can be discovered.

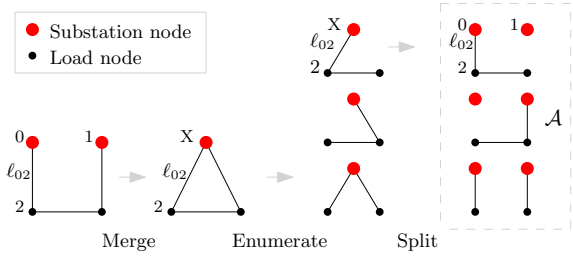

Figure 1. Rooted spanning forest enumeration process.

Due to the operational constraints, the agent must choose configurations that would lead to safe operation of the grid. As a result, many of the actions $a^r \in \mathcal{A}$ cannot be selected under certain injection pattern. Therefore, we reduce the action space $\mathcal{A}$ to include only those configurations that appeared in the historical data set. This allows the agent to avoid selecting unacceptable network configurations. However, this will limit the potential of discovering the optimal control policies.

C. Operational Experience Augmentation

One major drawback of the existing deep reinforcement learning algorithms is the poor sample efficiency. To improve the performance of our proposed deep Q-learning algorithm for dynamic DNR problem, we propose an innovative technique to generate reliable synthetic operational experience data from historical operational data set.

We propose a three-step algorithm to create a set of synthetic operational experiences $\tilde{D}_{\tilde{H}} \doteq \{\tilde{e}_1, ..., \tilde{e}_{\tilde{H}}\}$ where $\tilde{e}_t = (\tilde{S}_t = [\tilde{p}_t, \tilde{q}_t, \tilde{a}^r_{t-1}, t], \tilde{A}_t = \tilde{a}^r_t, \tilde{R}_{t+1} = r(\tilde{S}_t, \tilde{A}_t) + \lambda c(\tilde{S}_t, \tilde{A}_t), \tilde{S}_{t+1})$. The steps are 1) synthesizing the injection time series $\tilde{p}_t$ and $\tilde{q}_t$, 2) generating the network configuration

at each time step $\tilde{a}_t^r$, and 3) estimating the corresponding reward values $r(\tilde{S}_t, \tilde{A}_t) + \lambda c(\tilde{S}_t, \tilde{A}_t)$ for the data created in steps 1 and 2. Step 1 takes the historical load time series and outputs a new one. For example, we can either directly use the historical injection data or train an load time series model using historical data [20]. In step 2, we generate a sample path $\{a_t^r\}$ from a stochastic process defined on the sample space $\mathcal{A}$. In step 3, we estimate $r(\tilde{S}_t, \tilde{A}_t)$ and $c(\tilde{S}_t, \tilde{A}_t)$ for each time step t. The algorithms for estimating the network losses and voltage magnitudes are described below.

Two sets of regression models are trained on the historical data to estimate total network loss and nodal voltage magnitudes, respectively. For both sets of regression models, the input variables are the injection patterns and the network configurations. After the training, the reward $r(\tilde{S}_t, \tilde{A}_t) + \lambda c(\tilde{S}_t, \tilde{A}_t)$ can then be calculated based on the out-of-sample prediction of the regression models applied to the synthesized data points $\tilde{S}_t, \tilde{A}_t$. It has been shown that inaccurate rewards in training data can hurt the learning process. Therefore, we must determine if the estimated rewards are reliable and discard the ones which have high uncertainty.

We choose the Gaussian process (GP) [21] as the regression model to learn both the estimated values and their uncertainties. In the GP setting, the target y and the input vector x are modeled by the relationship $y = f(x) + \epsilon$ where ϵ represents the observation noise and is typically a zero mean Gaussian $\epsilon \sim \mathcal{N}(0, \sigma_\epsilon^2)$. f is a GP $f(x) \sim \mathcal{GP}(m(x), k(x, x'))$. If the mean function $m(x)$, the covariance function $k(x, x')$, and σ_ϵ^2 are known, then the probability distribution of any data $p(y|x)$ can be evaluated and the uncertainty is represented by the variance of $p(y|x)$. Typically, the mean and covariance functions of f are in some parametric families $m_{\theta_M}(x)$ and $k_{\theta_K}(x, x')$. For example, the constant mean function and the squared exponential covariance function are given by:

$$m_{\theta_M}(x) = C \quad k_{\theta_K}(x, x') = A^2 \exp\left(-\frac{\|x - x'\|_2^2}{2\ell^2}\right) \tag{8}$$

In this example, $\theta_M = \{C\}$ and $\theta_K = \{A, \ell\}$. The parameters θ_M, θ_K, and σ_ϵ^2 can be estimated by marginalizing the Gaussian process $\mathcal{GP}(m_{\theta_M}, k_{\theta_K})$ onto the training data points $\mathbf{x}$. That is, $\mathbf{y} \sim \mathcal{N}(\mu_\mathbf{x}, \Sigma_{\mathbf{xx}} + \sigma_\epsilon^2 I)$ where $\mu_\mathbf{x} = m_{\theta_M}(\mathbf{x})$ and $\Sigma_{\mathbf{xx}} = k_{\theta_K}(\mathbf{x}, \mathbf{x})$. Then we can perform maximum likelihood estimation of the parameters on this marginal distribution. Let the estimated parameters be $\hat{\theta}_M$, $\hat{\theta}_K$, and $\hat{\sigma}_\epsilon^2$. The posterior distribution of a testing instance $y^* = f(x^*) + \epsilon$ is again Gaussian, with the conditional mean and variance:

$$\hat{\mu}(y^*|x^*, \mathbf{x}, \mathbf{y}) = \hat{\mu}_{x^*} + \hat{\Sigma}_{x^*\mathbf{x}}(\hat{\Sigma}_{\mathbf{xx}} + \hat{\sigma}_\epsilon^2 I)^{-1}(\mathbf{y} - \hat{\mu}_\mathbf{x}) \tag{9}$$

$$\hat{\sigma}^2(y^*|x^*, \mathbf{x}, \mathbf{y}) = \hat{\sigma}_{x^*}^2 + \hat{\sigma}_\epsilon^2 - \hat{\Sigma}_{x^*\mathbf{x}}(\hat{\Sigma}_{\mathbf{xx}} + \hat{\sigma}_\epsilon^2 I)^{-1}\hat{\Sigma}_{\mathbf{x}x^*} \tag{10}$$

where $\hat{}$ means that the quantity is obtained by using the parameter estimates $\hat{\theta}_M$ and $\hat{\theta}_K$. $\hat{\sigma}^2(y^*|x^*, \mathbf{x}, \mathbf{y})$ is not quite the model uncertainty due to the lack of information about θ_M,θ_K, and σ_ϵ^2 [22]. An improved version is given by [22]:

$$\mathring{\sigma}^2(y^*|x^*, \mathbf{x}, \mathbf{y}) = \hat{\sigma}^2(y^*|x^*, \mathbf{x}, \mathbf{y}) + g^T M^{-1} g \tag{11}$$

where $g = \frac{\partial}{\partial \theta_M}[m_{\theta_M}(x^*) - m_{\theta_M}(\mathbf{x})^T(\hat{\Sigma}_{\mathbf{xx}} + \hat{\sigma}_\epsilon^2 I)^{-1}\hat{\Sigma}_{\mathbf{x}x^*}]$ and $M = \frac{\partial m_{\theta_M}(\mathbf{x})}{\partial \theta_M}(\hat{\Sigma}_{\mathbf{xx}} + \hat{\sigma}_\epsilon^2 I)^{-1}\left[\frac{\partial m_{\theta_M}(\mathbf{x})}{\partial \theta_M}\right]^T$. Now, $\hat{\mu}(y^*|x^*, \mathbf{x}, \mathbf{y})$ and $\mathring{\sigma}^2(y^*|x^*, \mathbf{x}, \mathbf{y})$ represent the estimated target and its uncertainty. In the dynamic DNR problem, each x represents an injection pattern and a radial configuration and each y represents the corresponding network loss or a voltage magnitude. If the uncertainty of the target estimate $\mathring{\sigma}^2(y^*|x^*, \mathbf{x}, \mathbf{y})$ is larger than some threshold, then the synthetic data generated (x^*, y^*) will be discarded. In this paper, the threshold is heuristically set to be $3 \cdot [\mathrm{std}(\mathring{\boldsymbol{\sigma}} - \mathrm{avg}(\mathring{\boldsymbol{\sigma}}))]$ where $\mathring{\boldsymbol{\sigma}}$ is the set of uncertainty estimates for all y^*.

D. Algorithm Summary

The technical methods of RL based dynamic DNR is summarized in Algorithm.1. The algorithm learns a control policy based on historical operation data. To address the sample efficiency problem of RL algorithms, we create a synthetic operation data set and combine it with the historical operation data. The synthetic data set is generated by combining the injection patterns $\tilde{p}_t, \tilde{q}_t$, network configurations $\tilde{a}_t^r$, and estimates of the corresponding network losses $\tilde{p}_t^l$ and voltages $\tilde{v}_{it}$ using GP models. The synthetic samples with large estimation uncertainties are removed from the augmented data set. The parameters of GP models are trained using the maximum likelihood framework with the historical operation data. Both the historical and augmented data are converted to experiences e_t and stored in the replay buffer D. We then use stochastic gradient descent to train the Q-learning agent. In each training iteration, the algorithm samples a random minibatch of experiences from D and updates the value network parameters θ^Q. The target network parameters θ^{Q-} are updated every C training iterations.

IV. Numerical Study

A. Experimental Data Description

The 16-bus distribution test feeder presented in [23, Example 1] is used in this study. The line impedances, RCSs, reference voltage (1.0 p.u.), and complex power base (100MVA) of [23] are kept unchanged. Applying the rooted spanning forest enumeration procedure in Section III.B, a total of 190 radial configurations are found. We replace the original static load data in [23] with 26 weeks of aggregated hourly real-world smart meter data of residential and commercial customers taken from a 12 KV distribution feeder.

The real-world smart meter data are reprocessed as follows. First, each nodal injection in the 16-bus feeder is set to be the aggregated consumption of a group of randomly selected customers. We assume a constant power factor for each node. Then, we scale this aggregated consumption by a common factor β (i.e., $(p_t, q_t) \mapsto (\beta p_t, \beta q_t)$ for all t) in order to create a realistic feeder loading level. β is chosen such that the resulting total line loss under βp_t and βq_t is roughly 1.5% of the total demand [24]. Next, we select 83 medium to low line loss configurations from a total of 190 feasible ones. A sample path of 26 weeks with hourly granularity is then generated from a Markov chain defined on those 83 configurations with

Algorithm 1 RL based Dynamic DNR with Operation Experience Augmentation

Input: Historical injections $\{p_t, q_t\}_{t=1}^{H}$, losses $\{p_t^l\}_{t=1}^{H}$, voltages $\{v_{it}\}_{t=1}^{H}, i \in N^v$, and configurations $\{a_t^r\}_{t=0}^{H}$.
Output: Policy π: $S_t \mapsto \text{argmax}_a Q(S_t, a, \theta^Q)$.

Train $1 + |N^v|$ Gaussian process models using $\{p_t, q_t, p_t^l, v_{it}, a_t^r\}_{t=1}^{H}$
Create synthetic injection data $\{\tilde{p}_t, \tilde{q}_t\}_{t=1}^{\tilde{H}}$ and configurations $\{\tilde{a}_t^r\}_{t=0}^{\tilde{H}}$
Estimate $\tilde{p}_t^l, \tilde{v}_{it}, i \in N^v$ using (9) and $\mathring{\sigma}^2(\tilde{p}_t^l), \mathring{\sigma}^2(\tilde{v}_{it}), i \in N^v$ using (11)
Create the experience set $D_H = \{e_1, \cdots, e_H\}$ using $\{p_t, q_t, p_t^l, v_{it}, a_t^r, a_0^r\}_{t=1}^{H}$
Create the augmented experience set $\tilde{D}_{\tilde{H}} = \{\tilde{e}_1, \cdots, \tilde{e}_{\tilde{H}}\}$ using $\{\tilde{p}_t, \tilde{q}_t, \tilde{p}_t^l, \tilde{v}_{it}, \tilde{a}_t^r, \tilde{a}_0^r\}_{t=1}^{\tilde{H}}$
for $t = 1, \cdots, \tilde{H}$ **do**
    if any $\mathring{\sigma}^2(\tilde{p}_t^l), \mathring{\sigma}^2(\tilde{v}_{it}), i \in N^v >$ threshold **then**
        $\tilde{D}_{\tilde{H}} \leftarrow \tilde{D}_{\tilde{H}} \setminus \{\tilde{e}_t\}$
$D \leftarrow D_H \cup \tilde{D}_{\tilde{H}}$
for $i = 1, \cdots, M$ **do**
    Sample a random minibatch $\{e_t\}^B \subset D$
    Perform a gradient descent step on $L(\theta^Q)$ with respect to θ^Q, where

$$L(\theta^Q) = \frac{1}{B} \sum_{e \in \{e_t\}^B} (r + \gamma \max_{a'} Q(s', a'; \theta^{Q-}) - Q(s, a; \theta^Q))^2$$

    if $\text{mod}(i, C) == 0$ **then**
        Update the target Q network $\theta^{Q-} \leftarrow \theta^Q$

transition probability $p_{ii} = 0.9$ and $p_{ij} = p_{ik}$. Finally, we find the power flow solutions for all hours in the 26-week period and record the total line losses as well as the voltage magnitude measurements at bus 7, 12, and 16 of the network to form the historical operational data set.

B. Setup of the Reward Function

By (1) and (5), the reward function is defined as the sum of negative costs of line losses (denoted as $C^l(s, a)$), switching actions (denoted as $C^s(s, a)$), and a weighted constraint violation term $\lambda c(s, a)$. C^l equals to the product of a fixed retail electricity price and the network losses. We set the retail electricity price at \$0.13/kWh. C^s equals the product a fixed cost per switching and the number of switching actions.

The fixed cost per switching is determined as follows. First, the lifetime cost of a sectionalizing switch can be calculated as the summation of the equipment cost, installation cost, and maintenance costs over its useful life. [25]. The sum of equipment cost and installation cost is assumed to be \$4,700. The useful life and annual maintenance cost of a switch are set to be 15 years and \$94. Thus, the lifetime cost of a sectionalizing switch is \$6,110. If we assume that the number of operations of a switch over its lifetime is 657 [26], then the fixed cost per switching is approximately \$4.6.

The upper and lower bounds $\bar{v}, \underline{v}$ for the voltage violation term is chosen as 1.1 and 0.9 p.u., respectively. λ is chosen to be \$0.13/kWh $\times$ 100MVA = \$13,000/p.u.

C. Performance of Operational Experience Augmentation

In this subsection, we validate the quality of the synthetic operational experience data generated by our proposed GP based model. In particular, the quality of estimated network losses under the augmented network configuration and injection patterns will be evaluated. Recall that we have 26 weeks of historical data, which are divided into training data (D_H) and testing data. The first 25 weeks of historical data are chosen as the training data and the data of the last week are chosen as the testing data.

We then create a 25-week synthetic operational data set $\tilde{D}_{\tilde{H}}$ as follows. First, we generate a 25-week sample path of network configurations from a Markov chain defined on those configurations that appeared in the training data set, with transition probability $p_{ii} = 0.8$ and $p_{ij} = p_{ik}$. We then estimate the network losses for this new sequence of configurations under the injection patterns of the first 25 weeks of historical data set. For network loss estimation, we compare the GP model in Section III.C with the Monte Carlo (MC) dropout neural network [27], which is shown to be equivalent to a Bayesian approximation of a GP. When building the GP model, the mean function is chosen to be zero and the covariance function is chosen to be the same as in (8). Both the GP and the MC dropout model are trained with the first 25 weeks of historical operational data. We apply the trained model to the 1-week testing data set and the 25-week synthetic operation experience data set. Fig.2 shows the performance of network losses prediction for the two models under 50 samples of both the testing data set and the synthetic data set. As shown in the figure, compared to the MC dropout model, the GP model is more accurate in predicting network losses.

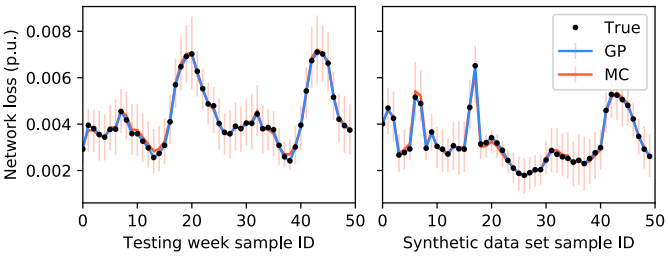

Figure 2. Performance of out-of-sample predictions for network losses.

Although GP model produces fairly accurate predictions, it occasionally leads to large error for some network configurations and injection patterns as shown by the orange curve in Fig.3. Fortunately, the uncertainty estimates of the GP model represented by the blue curve in Fig.3 correlates very well with the estimation error. This suggests our proposed strategy of removing the samples with large uncertainty estimates improves the quality of the augmented operational data set.

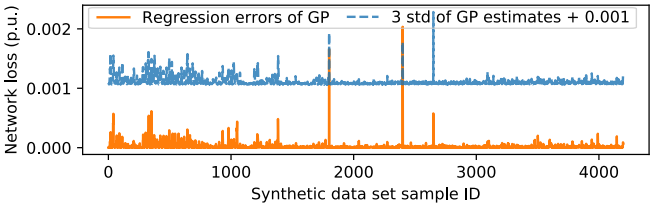

Figure 3. Regression errors versus uncertainty estimates of the GP model.

D. Performance of Deep Q-Learning Algorithms

In this subsection, we compare the performance of three deep Q-learning algorithms with two benchmarks. In the first benchmark, global optimal solution of the dynamic DNR problem is obtained by dynamic programming with perfect knowledge of the network parameters and future injection patterns. The second benchmark simply uses the historical network configurations in the data set. The first Q-learning algorithm is trained using only historical data. The second Q-learning algorithm is trained with both historical and synthetic experiences, where the network losses are estimated based on the GP model. The third deep Q-learning algorithm is trained with both historical and synthetic operational experiences, where the network losses are obtained by power flow studies assuming perfect knowledge of the network parameters.

We divide the 26-week historical data set into a 25-week training (D_H) data set and a 1-week testing data set. The 25-week synthetic operational experience data set $\tilde{D}_{\tilde{H}}$ is generated in the same way as in Section IV.C. During the training iterations, we periodically save the parameters of the neural network and test its performance on the testing data set. The performance of the three Q-learning algorithms and two benchmarks are shown in Fig.4. The left subfigure shows the minimum voltage magnitude over all metered nodes and all hours. The right subfigure shows the total operational cost. For the three deep Q-learning algorithms, the average, the 10th, and the 90th percentile of the results from 10 independent runs are depicted.

Compared to the network configurations in the testing week of the historical data, the deep Q-learning algorithm quickly learned how to reduce the operational cost in a dynamic DNR problem. When we augment the historical operational experiences with synthetic operational data, then the operational cost of the deep Q-learning algorithm further reduces and the minimum voltage magnitudes get even closer to the nominal voltage values. As the learning process proceeds, the performance of the deep Q-learning algorithms with augmented operational experiences approaches that of the global optimal solution. Note that the Q-learning agents achieved these results without knowing the actual network parameters or future power injection patterns. It can also be seen from the figure that the orange curve almost coincides with the green curve, suggesting that the network losses estimated by our proposed GP model are almost as good as that of the power flow solutions with perfect network parameter information.

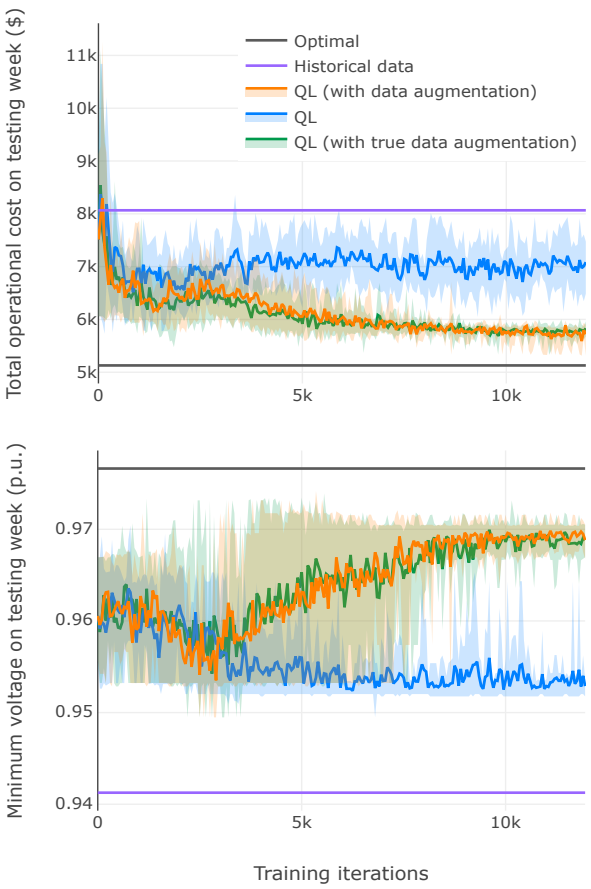

Figure 4. Performance of Q-learning. Hyperparameters of the neural network: 2-layer feed-forward (hidden: 600, output: 190); activation function: ReLU; optimizer: Adam; batch size: 64; discount factor γ: 0.95; update steps C: 30.

We conclude the numerical study by showing that similar results can be obtained without extensive tuning of hyperparameters, which is crucial for practical applications. We demonstrate this by showing that the performance of the Q-learning algorithm is relatively consistent under different hyperparameter settings. The following combinations of hyperparameters are tested: batch size $B \in \{32, 64, 128, 256\}$, number of hidden layers $L \in \{1, 2\}$, number of hidden neurons $H \in \{300, 400, 500, 600\}$, and number of steps the target Q network's parameters are updated $C \in \{30, 60, 90, 120\}$. We generate a Taguchi's orthogonal array for these hyperparameter combinations and report the results in Table.I. Each calculated cost represents the average of 5 independent runs for Q-learning with operational data augmentation. Compared to the historical operational cost and the optimal cost, the operational cost of the Q-learning algorithm under different hyperparameter settings are quite consistent.

V. Conclusion and Discussion

This paper presents a reinforcement learning based algorithm to solve the dynamic distribution network reconfiguration problem without accurate network parameter information.

Table I
OPERATIONAL COSTS WITH VARIOUS HYPERPARAMETERS

Original cost: \$ 8066.7					Optimal cost: \$ 5128.8				
B	H	C	L	QL cost (\$)	B	H	C	L	QL cost (\$)
32	300	30	1	5752.9	128	300	90	1	5490.6
32	400	60	1	5686.2	128	400	120	1	5647.4
32	500	90	2	5608.8	128	500	30	2	5441.0
32	600	120	2	5464.3	128	600	60	2	5396.7
64	300	60	2	5523.9	256	300	120	2	5480.4
64	400	30	2	5456.8	256	400	90	2	5487.8
64	500	120	1	5579.3	256	500	60	1	5652.2
64	600	90	1	5507.9	256	600	30	1	5531.3

The proposed framework first formulates the dynamic DNR problem as a Markov decision process, then learns the approximated optimal action-value function with a neural network. The optimal network configuration is selected to be the action that yields the highest action-value. A novel synthetic operational experience data generation technique based on the Gaussian process is developed to improve the performance of Q-learning algorithms. Simulation results show that the proposed Q-learning algorithm successfully reduces the operational cost of the network under various hyperparameter settings.

The primary limitation of the proposed framework is scalability. When the number of loops in the all-switch-closed network is large, enumerating all radial configurations can be impractical. For example, the 33-bus system [28] has 50,751 radial configurations. Learning to reconfigure this network can be slow and requires a large amount of historical data. We plan to improve the scalability of our algorithm in future studies.

REFERENCES

[1] R. A. Jabr, R. Singh, and B. C. Pal, "Minimum loss network reconfiguration using mixed-integer convex programming," *IEEE Transactions on Power Systems*, vol. 27, no. 2, pp. 1106–1115, May 2012.

[2] N. Yu, S. Shah, R. Johnson, R. Sherick, M. Hong, and K. Loparo, "Big data analytics in power distribution systems," in *2015 IEEE Power Energy Society Innovative Smart Grid Technologies Conference (ISGT)*, Feb 2015, pp. 1–5.

[3] Y. Gao, P. Wang, and N. Yu, "A distributed algorithm for distribution network reconfiguration," in *2018 China International Conference on Electricity Distribution (CICED)*. IEEE, 2018, pp. 1730–1734.

[4] F. Capitanescu, L. F. Ochoa, H. Margossian, and N. D. Hatziargyriou, "Assessing the potential of network reconfiguration to improve distributed generation hosting capacity in active distribution systems," *IEEE Transactions on Power Systems*, vol. 30, no. 1, pp. 346–356, Jan 2015.

[5] B. Novoselnik and M. Baotić, "Dynamic reconfiguration of electrical power distribution systems with distributed generation and storage," *IFAC-PapersOnLine*, vol. 48, no. 23, pp. 136–141, 2015.

[6] F. V. Dantas, D. Z. Fitiwi, S. F. Santos, and J. P. S. Catalão, "Dynamic reconfiguration of distribution network systems: A key flexibility option for res integration," in *2017 IEEE International Conference on Environment and Electrical Engineering and 2017 IEEE Industrial and Commercial Power Systems Europe (EEEIC / I&CPS Europe)*, June 2017, pp. 1–6.

[7] W. Jianzhong and Y. Yixin, "Global algorithm to time-varying reconfiguration for operation cost minimization," *Proceedings of the CSEE*, 2003, (in Chinese).

[8] E. A. Feinberg, J. Hu, and K. Huang, "A rolling horizon approach to distribution feeder reconfiguration with switching costs," in *2011 IEEE International Conference on Smart Grid Communications (SmartGridComm)*, Oct 2011, pp. 339–344.

[9] H. Yang, Y. Peng, and N. Xiong, "Gradual approaching method for distribution network dynamic reconfiguration," in *2008 Workshop on Power Electronics and Intelligent Transportation System*, Aug 2008, pp. 257–260.

[10] M. Mosbah, S. Arif, R. D. Mohammedi, and A. Hellal, "Optimum dynamic distribution network reconfiguration using minimum spanning tree algorithm," in *2017 5th International Conference on Electrical Engineering - Boumerdes (ICEE-B)*, Oct 2017, pp. 1–6.

[11] L. Xu, R. Cheng, Z. He, J. Xiao, and H. Luo, "Dynamic reconfiguration of distribution network containing distributed generation," in *2016 9th International Symposium on Computational Intelligence and Design (ISCID)*, vol. 1, Dec 2016, pp. 3–7.

[12] A. E. Milani and M. R. Haghifam, "An evolutionary approach for optimal time interval determination in distribution network reconfiguration under variable load," *Mathematical and Computer Modelling*, vol. 57, no. 1-2, pp. 68–77, 2013.

[13] W. Wang, N. Yu, B. Foggo, J. Davis, and J. Li, "Phase identification in electric power distribution systems by clustering of smart meter data," in *2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE, 2016, pp. 259–265.

[14] W. Wang, N. Yu, and Z. Lu, "Advanced metering infrastructure data driven phase identification in smart grid," *GREEN 2017 Forward*, p. 22, 2017.

[15] B. Foggo and N. Yu, "A comprehensive evaluation of supervised machine learning for the phase identification problem," *World Acad. Sci. Eng. Technol. Int. J. Comput. Syst. Eng*, vol. 12, no. 6, 2018.

[16] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[17] 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, p. 529, 2015.

[18] A. Nair, P. Srinivasan, S. Blackwell, C. Alcicek, R. Fearon, A. De Maria, V. Panneershelvam, M. Suleyman, C. Beattie, S. Petersen, S. Legg, V. Mnih, K. Kavukcuoglu, and D. Silver, "Massively parallel methods for deep reinforcement learning," in *Intl. Conf. on Machine Learning*, Lille, France, 2015.

[19] D. E. Knuth, *The art of computer programming, volume 4A: combinatorial algorithms, part 1*. Pearson Education India, 2011.

[20] J. W. Taylor, "Short-term electricity demand forecasting using double seasonal exponential smoothing," *Journal of the Operational Research Society*, vol. 54, no. 8, pp. 799–805, 2003.

[21] C. E. Rasmussen, "Gaussian processes in machine learning," in *Summer School on Machine Learning*. Springer, 2003, pp. 63–71.

[22] J. Wågberg, D. Zachariah, T. B. Schön, and P. Stoica, "Prediction performance after learning in gaussian process regression," in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS)*, Fort Lauderdale, FL, USA, 4 2017.

[23] C.-T. Su and C.-S. Lee, "Network reconfiguration of distribution systems using improved mixed-integer hybrid differential evolution," *IEEE Transactions on Power Delivery*, vol. 18, no. 3, pp. 1022–1027, July 2003.

[24] Michael D Simms, "The National Commission for Energy State Regulation of Ukraine Energy, The Public Utilities Commission of Ohio-Distribution System Losses," 2013, Duke Energy®.

[25] A. Abiri-Jahromi, M. Fotuhi-Firuzabad, M. Parvania, and M. Mosleh, "Optimized sectionalizing switch placement strategy in distribution systems," *IEEE Transactions on Power Delivery*, vol. 27, no. 1, pp. 362–370, Jan 2012.

[26] "The Reliability and Power Quality Performance of Overhead Lines – with reference to the Electrical Properties of Wood and Covered Conductors," https://espace.library.uq.edu.au/view/UQ:9818/saha-mat-dist200.pdf, accessed: 2018-09-10.

[27] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," in *international conference on machine learning (ICML)*, 2016.

[28] M. E. Baran and F. F. Wu, "Network reconfiguration in distribution systems for loss reduction and load balancing," *IEEE Transactions on Power Delivery*, vol. 4, no. 2, pp. 1401–1407, April 1989.