Accelerated Deep Reinforcement Learning Based Load Shedding for Emergency Voltage Control

Renke Huang, Senior Member, IEEE, Yujiao Chen, Tianzhiyi Yin, Xinya Li, Ang Li, Jie Tan, Quhia Huang, Member, IEEE,

Abstract—Load shedding has been one of the most widely used and effective emergency control approaches against voltage instability. With increased uncertainties and rapidly changing operational conditions in power systems, existing methods have outstanding issues in terms of either speed, adaptiveness, or scalability. Deep reinforcement learning (DRL) was regarded and adopted as a promising approach for fast and adaptive grid stability control in recent years. However, existing DRL algorithms show two outstanding issues when being applied to power system control problems: 1) computational inefficiency that requires extensive training and tuning time; and 2) poor scalability making it difficult to scale to high dimensional control problems. To overcome these issues, an accelerated DRL algorithm named PARS was developed and tailored for power system voltage stability control via load shedding. PARS features high scalability and is easy to tune with only five main hyperparameters. The method was tested on both the IEEE 39-bus and IEEE 300-bus systems, and the latter is by far the largest scale for such a study. Test results show that, compared to other methods, PARS has much better computational efficiency, and excellent scalability and generalization capability.

Index Terms—Deep reinforcement learning, Voltage stability, Load shedding, Augmented random search

I. INTRODUCTION

A. Motivation

Bulk power systems are facing increasing risks of voltage stability with greater presence of dynamic loads and expanding integration of inverter-based resources that lead to lower system strength and limited reactive capability during disturbances. Furthermore, tripping of distributed and centralized inverter-based resources during large transmission disturbances could deteriorate voltage stability and lead to voltage collapse or blackout [1]. Load shedding has been one of the most widely used and effective emergency control approaches to counteract voltage instability, in particular short-term voltage instability [2], [3]. However, existing methods have outstanding issues in terms of either speed, adaptiveness, or scalability. Thus, major enhancements of voltage control schemes are much needed. This paper focuses on developing an accelerated deep reinforcement learning (DRL)-based control method to make load shedding for emergency voltage control fast, adaptive, and scalable.

To date, it takes extensive time (ranging from days to even months) to train a practically good control policy for complex power systems using existing state-of-the-art DRL algorithms [4]. Accelerated training not only makes DRL more practical for real-world applications, but also brings significant benefits, including: 1) shorter experiment turnaround time facilitating better control design [5]; 2) overcoming the operational challenges associated with increasing uncertainties by enabling DRL training closer to real-time situations (e.g., moving from days ahead to hours ahead); and 3) the ability to update the emergency control schemes more frequently whenever necessary, which can enhance the adaptiveness and effectiveness of emergency control schemes during fast-changing events such as hurricanes and cascading outages.

B. Related Work

In designing a load shedding control scheme, the time, location, and amount are important and closely related aspects of load shedding against voltage instability [2]. Past efforts in determining these aspects for load shedding-based emergency control can be roughly categorized into rule-based, measurement-based, algorithmic decision-based, and learning-based (including data-driven) approaches.

Rule-based approach: Most existing load shedding control designs deployed in the power industry are rule based. A simple scheme relies on simple rules like “if voltage drops below some threshold $V_{th}$ for some duration $\tau$, shed some power $\Delta P$” [3]. While some enhancements to simple rule-based methods were proposed [6], the main issues of such methods are lack of adaptiveness and optimality.

Algorithmic decision-based approach: Security-constrained alternating current power flow optimization [7] was proposed for grid emergency control. Another widely used algorithmic approach is model-predictive control [8]. One serious limitation of this type of method is poor scalability due to computational complexity. In addition, since they rely on a system model, these methods are susceptible to model inaccuracies.

Measurement-based approach: In recent years, methods for real-time voltage control were developed by leveraging phasor measurement unit technologies and methodologies have been developed for tracking voltage behavior [9]–[11]. Without high-level coordination that would require some algorithmic decision-based or learning-based approaches, these methods...
mainly address local issues and may not be adequate to mitigate emergencies at the system level.

Learning-based approach: Learning-based (or data-driven) methods gained much attention and interest for grid control in both academia and industry in the past decade. A decision-tree-based approach was proposed for preventive and corrective control [12]. A hierarchical, extreme learning machine-based method for load shedding against fault-induced delayed voltage recovery (FIDVRR) events was developed in [13]. An important learning-based approach for controlling dynamic systems is reinforcement learning (RL). There is a significant number of previous efforts utilizing conventional RL methods such as Q-learning for many different power system control applications and areas [14], [15]. Yet conventional RL approaches have serious limitations in terms of processing high-dimensional observation and action spaces as well as difficulty in large-scale training. Recent outstanding advancements in integrating deep-learning techniques with RL overcome many of these limitations. DRL has led to many breakthroughs in controlling complex systems, particularly in games, robotic control, and autonomous driving. In our previous work [4], we adapted a popular DRL algorithm called deep Q-network (DQN) and achieved adaptive emergency control schemes for both generator dynamic breaking and under-voltage load shedding (UVLS). Another DRL algorithm, deep deterministic policy gradient (DDPG), was applied for emergency load shedding schemes in [16].

It should be noted that many state-of-the-art DRL algorithms such as DQN and DDPG used in [4], [16] are notoriously known for data sampling inefficiency, difficulty in scaling up the solutions, and time-consuming for hyperparameter tuning. However, the training time (including tuning) and scalability issues were not addressed in the previous power domain applications [4], [16]. Techniques for accelerating some state-of-the-art policy-gradient and value-based DRL algorithms, particularly by leveraging a combination of central and graphical processing units (CPUs and GPUs), were developed in [5], [17]. It should also be noted that these techniques and their implementations are mainly targeted and optimized for video game environments, which are much less computationally intensive compared to large and complex power system dynamic simulations. For example, video games can be efficiently simulated and processed in GPUs, but this is not the case for power system dynamic simulations. This means that existing accelerated techniques and frameworks [5], [17] are generally not suitable for power system control applications.

C. Contributions

The main contributions of this paper include:

- Novel application and adaptation of Augmented Random Search (ARS) algorithm coupled with forward neural network (FNN) and the long short-term memory (LSTM) network [18] for emergency voltage control via adaptive load shedding.
- Accelerating ARS algorithms for grid control through parallelizing the ARS algorithms and power system dynamic simulations systematically, leading to 100X speedup training on the IEEE 300-bus system.

D. Organization

The rest of the paper is organized as follows: Section II introduces DRL and the problem formulation. Section III presents the ARS algorithm and its enhancements and techniques for accelerating the algorithm for grid control. Test cases and results are shown in Section IV. Conclusions and future work are provided in Section V.

II. DRL-BASED LOAD SHEDDING FOR EMERGENCY VOLTAGE CONTROL

This section presents load shedding for emergency voltage control, an introduction to DRL, and the problem formulation.

A. Emergency Voltage Control via Load Shedding

Among the measures of emergency voltage control, load shedding is well known as an effective countermeasure against voltage instability [19]. It has been widely adopted in the industry, mostly in the form of rule-based UVLS. The UVLS relays are usually employed to shed load demands at substations in a step-wise manner if the monitored bus voltages fall below the predefined voltage thresholds. ULVS relays have a fast response, but do not have communication or coordination between other substations, leading to unnecessary load shedding [20] at affected substations.

As pointed out in the introduction, there are three key factors for load shedding: time, location, and amount. To optimally determine these three factors simultaneously, one has to solve a highly non-linear, non-convex, optimal decision-making problem. A detailed mathematical formulation as a constrained optimization control problem and its conversion to a Markov decision process (MDP) formulation can be found in the recent work of the authors [4]. As this paper focuses on a DRL-based solution to this problem, a brief introduction to DRL is provided below, followed by a problem formulation using MDP.

B. Deep Reinforcement Learning

A RL problem can be defined as policy search in a (partially observable) MDP defined by a tuple $(S, A, P, r)$ [21]. The state space $S$ and action space $A$ could be continuous or discrete. In this paper, both of them are continuous. The environment transition function $P : S \times A \times S \rightarrow \mathbb{R}$ is the probability density of the next state $s_{t+1} \in S$ given the current state $s_t \in S$ and action $a_t \in A$. At each interaction step, the environment returns a reward $r : S \times A \rightarrow \mathbb{R}$. The standard RL objective is the expected sum of discounted rewards. The goal of an agent is to learn a policy $\pi^*(s_t, a_t)$ that maximizes the objective.

DRL is a combination of RL and deep-learning technologies. The capabilities of high dimensional feature extraction and non-linear approximation from deep learning makes it possible for DRL to directly use the raw state-space representations and train policies for complex systems and tasks.
in a more effective and efficient way. When neural networks are used as function approximation for the policy of DRL, gradients need to be calculated at some point to update the network weights during training. However, when applied for the power grid emergency voltage control problem, gradient is not easy to estimate from a sophisticated power system simulation. Furthermore, the fact that the planning stage has not been able to scale reliably to high dimensions also contributes to the poor scalability of model-based approaches [22]. Recently, there are several derivative-free methods such as ARS [18] and natural evolution strategies [23] that have been developed as competitive and highly scalable alternatives to other gradient-based DRL algorithms.

Fig. 1 shows comparisons of different classes of RL algorithms and alternatives in terms of sampling efficiency and computational scalability. Sampling efficiency means how much data need to be collected to train the RL algorithm. Computational scalability is mainly related to parallelization and speeding up the training process. Generally speaking, the fewer sensitive hyperparameters one RL algorithm has, the easier (and less time) it takes to tune them to achieve good performance. These three measures are the main factors in determining the proper RL algorithms for particular applications. The ARS algorithm generally is easier to scale in distributed settings, and thus has better computational efficiency [18], while its shortcoming is the relatively poor sampling efficiency. As high-performance grid and parallel simulations [24] can be leveraged to fully speed up producing plentiful data to train the ARS policy offline, the weakness in sampling efficiency can be offset by its scalability advantage, provided that sufficient computing resources are available. It should be noted that the small number of sensitive hyperparameters of ARS means it is easier to tune compared to other existing DRL methods. More details of ARS will be discussed in the next section. In light of these properties, we have adopted ARS and further enhance and accelerate it to achieve the object of designing a fast, adaptive, and scalable DRL-based load shedding scheme in this paper.

C. MDP Formulation for Load Shedding

State: The observed system variables $O_t$ at time $t$ include voltage magnitudes at monitored buses (denoted as $V_i$) as well as the percentage of load still remaining at controlled buses (denoted as $P_{Dt}$). To capture the dynamics of the voltage change, the most recent observed states could be stacked with some historical state records and treated as the actual state for the agent at time $t$, i.e., $s_t = (O_{t-N_t-1}, \cdots, O_t)$.

Action: The control action at each controlled load bus is to shed a percentage (within [0, 20%] in this paper) of the total load at each action time step. Thus, the action space is continuous with a range of [-0.2, 0] (minus means shedding) for each action bus.

State transition: The state transition is deterministic and governed by power system dynamics that are defined by a set of differential and algebraic equations [4].

Reward: The basic principle of designing the reward function is to guide the agent to meet the transient voltage recovery criterion that is defined to evaluate the system voltage recovery. Without loss of generality, we referred to the standard proposed in [25] and shown in Fig. 2. After fault clearance, the standard requires that voltages should return to at least 0.8, 0.9, and 0.95 p.u. within 0.33, 0.5, and 1.5 s. Accordingly, the reward $r_t$ at time $t$ is defined as follows:

$$
{r_t} = \begin{cases} 
-1000, & \text{if } V_i(t) < 0.95, \quad t > T_{pf} + 4 \\
0.1, & \text{if } T_{pf} < t < T_{pf} + 0.33 \\
0.2, & \text{if } T_{pf} + 0.33 < t < T_{pf} + 0.5 \\
0.5, & \text{if } T_{pf} + 0.5 < t < T_{pf} + 1.5 \\
1.0, & \text{if } T_{pf} + 1.5 < t 
\end{cases}
$$

where $T_{pf}$ is the time instant of fault clearance. The above reward function has three parts: (1) total bus voltage deviation below the standard voltage thresholds shown in Fig. 2, where $V_i(t)$ is the bus voltage magnitude for bus $i$ in the power grid; (2) total load shedding amount, where $\Delta P_j(t)$ is the load shedding amount in p.u. at time step $t$ for load bus $j$; and (3) invalid action penalty $u_{ivld}$ if the DRL agent still provides load shedding action when the load at a specific bus has already been shed to zero in the previous time step when the system is within normal operation. The weight factors for the above three parts are $c_1$, $c_2$, and $c_3$. Note that the reward function will be set to a large negative number (-1000) if any bus voltage is below 0.95 p.u. 4 s after the fault is cleared.

III. ARS ALGORITHM AND ITS ENHANCEMENTS

The ARS algorithm was originally proposed in [18] as a competitive alternative to conventional model-free DRL algorithms. In this paper, we have enhanced, accelerated, and tailored it for power system voltage stability control via load shedding.

![Fig. 1. Comparison of different RL algorithms](image1.png)

![Fig. 2. Transient voltage recovery criterion for transmission system](image2.png)
A. ARS algorithm

Different from existing model-free DRL algorithms that use action-space exploration, ARS performs parameter-space exploration and estimates the gradient of the returns using sampled rollouts, thus back-propagation is not needed. Compared to existing DRL algorithms (i.e., TRPO, DDPG, PPO, A2C, and SAC), Mania et al. [18] demonstrated that ARS can achieve comparable or even better performance in robotic continuous control problems while taking less wall clock time to train. Furthermore, in contrast to many gradient-based DRL algorithms such as DDPG and PPO having more than 20 hyperparameters, ARS has only five main hyperparameters, i.e., $\alpha$, $\upsilon$, $N$, $b$, and $m$ in Algorithm 1, which makes it much easier for end users to achieve satisfactory control performance without extensive tuning.

The ARS algorithm employed in this paper is shown in Algorithm 1, which is modified based on [18]. To scale up the ARS algorithm for large-scale control problems and reduce the training time, we accelerate it by leveraging its inherent parallelism and implementing it on a high-performance computing platform. Details are discussed in subsection III-B.

The original ARS algorithm was proposed for linear control policies with the policy $\theta$ represented by a matrix [18]. Our test results in Section IV show that the linear policy representation does not perform well for highly non-linear power system voltage stability control problems. To overcome these shortcomings, we enhanced ARS by modeling policies with neural networks, namely the FNN and the LSTM network [26]. Details of deploying both the FNN and LSTM together with ARS are provided in subsection III-C.

B. Accelerating ARS Algorithms for Power System Control

To explore the parameter space of the control policy efficiently and be adaptive to multiple tasks (in this paper we define different fault scenarios in the power grid as multiple tasks, denoted by task set $T$), the ARS algorithm needs to perform a large number of power grid dynamic simulations (environment rollouts) by inferring with a sufficient number of different perturbed policies at each iteration of the training. Parallelizing power grid dynamic simulations in the ARS training plays a critical role for accelerating the training speed. ARS supports parallelism in steps 5 and 7 in Algorithm 1 by nature, the crucial part is how to implement it efficiently and effectively based on the requirements and special characteristics of power system dynamic simulation and control.

The parallel version of the ARS algorithm (named PARS) is implemented with the Ray framework [27], which supports task parallelism (via Ray remote functions) and actor-based computation (via Ray remote classes). The structure of the two-level parallelism is illustrated in Fig. 3, which includes a perturbation parallelism (policy level) and an environment rollout parallelism (task level), corresponding to steps 5 and 7 in Algorithm 1, respectively. The ARS learner is an actor at the top to delegate tasks and collect returned information, and controls the update of policy weights $\theta$. The learner communicates with subordinate workers and each of these workers is responsible for one or more perturbations (random search) of the policy weights. The ARS learner combines the results from each worker and updates the policy weights centrally based on the perturbation results from the top performing workers. The workers do not execute environment rollout tasks by themselves. They spawn a number of slave actors and assign these tasks to subordinate slave actors. Note that each worker needs to collect the rollout results from multiple tasks inferring with the same perturbed policy, and each slave actor is only responsible for one environment rollout with the specified task and perturbed policy sent by its up-level worker. For the environment rollouts, power system dynamic simulations are performed by RLG [28], which is an open source tool for developing and benchmarking RL algorithms for grid control.

### Algorithm 1 Modified ARS

1. **Hyperparameters:** Step size $\alpha$, number of policy perturbation directions per iteration $N$, standard deviation of the exploration noise $\upsilon$, number of top-performing perturbed directions selected for updating weights $b$, number of rollouts per perturbation direction $m$. Decay rate $\varepsilon$.
2. **Initialize:** Policy weights $\theta_0$ with small random numbers; the running mean of observation states $\mu_0 = 0 \in \mathbb{R}^n$ and the running standard deviation of observation states $\Sigma_0 = I_n \in \mathbb{R}^{n \times n}$, where $n$ is the dimension of observation states, the total iteration number $H$.
3. **for** iteration $t = 1,...,H$ **do**
4. **sample** $N$ random directions $\delta_1,...,\delta_N$ with the same dimension as policy weights $\theta$
5. **for** each $\delta_i(i \in [1,...,N])$ **do**
6. add $\pm$ perturbations to policy weights: $\theta_{t+1} = \theta_{t-1} + \upsilon \delta_i$ and $\theta_{t-1} = \theta_{t-1} - \upsilon \delta_i$
7. **do** total $2m$ rollouts (episodes) denoted by $R_{p,T}(\cdot)$ for different tasks $p$ sampled from task set $T$ based on the $\pm$ perturbed policy weights, calculate the average rewards of $m$ rollouts as the rewards for $\pm$ perturbations, i.e., $r_{t+1}$ and $r_{t-}$:

$$
\begin{align}
  r_{t+1} &= \frac{1}{m} R_{p,T}(\theta_{t+1}, \mu_{t-1}, \Sigma_{t-1}) \\
  r_{t-} &= \frac{1}{m} R_{p,T}(\theta_{t-1}, \mu_{t-1}, \Sigma_{t-1})
\end{align}
$$

8. During each rollout, states $s_{t,k}$ at time step $k$ are first normalized and then used as the input for inference with policy $\pi_{\theta_0}$ to obtain the action $a_{t,k}$, which is applied to the environment and new states $s_{t+1,k}$ is returned, as shown in (3). The running mean $\mu$ and standard deviation $\Sigma$ are updated with $s_{t,k+1}$

$$
\begin{align}
  s_{t,k+1} &= (s_{t,k} - \mu_{t-1})/\Sigma_{t-1} \\
  a_{t,k} &= \pi_{\theta_0}(s_{t,k}) \\
  s_{t+1,k} &\leftarrow P(s_{t,k}, a_{t,k})
\end{align}
$$

9. **end for**
10. sort the directions based on $\max[r_{t+1}, r_{t-}]$ and select top $b$ directions, calculate their standard deviation $\sigma_b$
11. update the policy weight:

$$
\theta_{t+1} = \theta_t + \frac{\alpha}{b \sigma_b} \sum_{i=1}^{b} (r_{t+1} - r_{t-}) \delta_i
$$

12. **Step size $\alpha$ and standard deviation of the exploration noise $\upsilon$ decay with rate $\varepsilon$: $\alpha = \varepsilon \alpha$, $\upsilon = \varepsilon \upsilon$
13. **end for**
14. **return** $\theta$
Fig. 3. Parallel architecture of the ARS algorithm and supports task-level parallelism.

Details for the data and information exchanged between the ARS learner, the workers, and the environment rollout actors are shown in Fig. 3 and described as follows. At the beginning of a training iteration $t$, the ARS learner distributes the policy weights $\theta_{t-1}$, the mean $\mu_{t-1}$, and the standard deviation $\Sigma_{t-1}$ of the observations from the previous iteration $t-1$ to the workers that are responsible for different policy weight perturbations. For each worker $i$, it distributes the $\mu_{t-1}$, $\Sigma_{t-1}$, as well as the specially perturbed policy weights $\theta_{t-1} \pm \nu \delta_i$ to its slave actors. Each slave actor performs single environment rollout for a different task $p \in T$ ($T$ is the set of tasks) by inferring with the perturbed policy, and sends the new mean $\mu_{p,t,i}$ and standard deviation $\Sigma_{p,t,i}$ of the observations as well as the reward $R_{p,t,i}$ back to its master worker. Upon receiving the rewards, means, and standard deviations of observations from all slave actors, each worker $i$ computes the average reward $r_{i,t}$, new mean $\mu_{i,t}$, and standard deviation $\Sigma_{i,t}$ for the observations from all its tasks $T$ and sends them, together with the perturbation $\theta_i$, back to the ARS learner. Once the ARS learner receives all the information from its workers, it updates the policy weights according to (4) in Algorithm 1 and the training continues to the next iteration.

C. FNN and LSTM for Modeling Policies

We propose an innovative method of integrating the FNN and LSTM together with ARS to significantly enhance the performance of ARS. Note that traditionally weights of neural networks are updated with back-propagation using gradient descents, whereas in our PARS algorithm the weights of neural networks are updated using (4) in Algorithm 1. FNN is a commonly used neural network model for mapping the observations to actions in DRL algorithms to learn the non-linearity of their relationship. A FNN with two hidden, fully-connected (FC) layers (shown in Fig. 4a) is used in this paper.

The main advantage of FNN is that its simple architecture makes it easy to train. On the other hand, lack of capability of storing historical memory makes it challenging for power system stability control applications because observed states at one step do not capture important system dynamic features such as voltage dip or recovery trend. One solution is to stack some recent history observations as the actual input to FNN. This will inevitably increase the dimension of the input, and thus size of the FNN. Furthermore, the number of history observations to be stacked becomes a hyperparameter, which needs to be tuned on a task basis. This leads us to also explore adding LSTM for modeling policies for enhancing the ARS.

LSTM is a type of recurrent neural network that is capable of learning long-term dependencies, as shown in Fig. 5. LSTM uses cell state to capture the long-term dependencies of the data, which is determined by three gates, namely the input gate, forget gate, and output gate. LSTM is adopted in our study to learn the temporal correlation of the voltage observations without manually stacking a certain number (kind of feature engineering) to reduce the dimension of the inputs (compared with FNN) and thus accelerating the algorithm. After the LSTM layer, fully connected neural network layers are added, as shown in Fig. 4(b).

IV. TEST CASES AND RESULTS

Our ARS training framework is deployed on a local high-performance computing cluster with a Linux operating system which comprises 520 nodes. Each node features dual-socket Intel Haswell E5-2670V3 CPU (12 cores per socket running at 2.3 GHz) with 64 GB DDR4 memory. We tested the performance of our PARS algorithm with different numbers of computing nodes and cores. Tests were performed with both IEEE 39-bus and IEEE 300-bus systems. The data of both test systems are publicly available in [28].

A. Performance Metrics

To evaluate the performance of the developed ARS algorithm with its enhancements, the following metrics were defined and investigated.

1) Metrics for training: We considered (a) computational time and (b) convergence rate. The total computational time at each training iteration was recorded and accumulated. Less execution time for each training iteration
was an indicator of higher computational efficiency. RL training is considered as converged when its learning curve gets flat with small variations (e.g., 2%). The convergence rate can be represented by an minimum iteration number at which the average reward reaches a stable value. The smaller the iteration number achieving a stable average reward, the faster the training converges.

2) **Metrics for testing:** (a) The average rewards the trained policy obtained on the testing task set that was different from the training task set; (b) total load shedding amount. Note that the reward defined in Eq. (1) measures the success of the load shedding controls based on the factor that the control should shed as little load as possible to recover the system voltage. The comparison of the rewards between ARS and the baseline method (UVLS) is presented in the following subsections.

**B. Test Case 1: IEEE 39-bus Test System**

We applied our proposed PARS algorithm on the IEEE 39-bus test system (details of the system can be found in [4]) to learn a closed-loop control policy for applying the load shedding at a load center including buses 4, 7, and 18 to avoid the FIDVR and meet the voltage recovery requirements shown in Fig. 2. We first trained the ARS algorithm with linear, FNN, and LSTM models for representing the control policy. Observations included voltage magnitudes at buses 4, 7, 8, and 18 as well as the remaining fractions of loads served by buses 4, 7, and 18. For the linear and FNN models, the last 10 recent observations were stacked and used as input for ARS, thus the dimension of the input was 70; while for the LSTM models, there was no need for stacking the observation states from previous time steps, and thus the dimension of the input was 7. The control action for buses 4, 7, and 18 at each action time step was a continuous variable from 0 (no load shedding) to -0.2 (shedding 20% of the initial total load at the bus).

During the training, the task set $T$ was defined as nine different tasks (fault scenarios). Each task began with a flat start of dynamic simulation. At 1.0 s of the simulation time a short-circuit fault was applied at bus 4, 15, or 21 with a fault duration of 0.0 s (no fault), 0.05 s, or 0.08 s and the fault was self-cleared. The task set $T$ defined with multiple fault locations and durations could guarantee the ARS algorithm interacted with the system with and without FIDVR conditions. Other parameter settings for the ARS algorithm are listed in Table I. With the setting of 16 directions for the policy-level perturbation and 9 tasks for task-level domain randomization, the proposed two-level parallelism scheme needed a minimum of 144 cores for fully parallelizing the computation tasks. As a result, eight computational nodes with maximum available cores of 192 were used for training. Figure 6(A) shows the reward achieved by ARS using the three different policy architectures, averaged over the five random seeds. The linear model has the lowest performance among the three models. The major performance difference between the FNN and LSTM models is the computational efficiency: the LSTM model helped reduce the total time by 50% compared with the FNN model, due to the smaller input dimension (from 70 to 7), while the convergence curves of both models behave in a very similar way as shown in Fig. 6(A).

**TABLE I**

| Baseline configuration for training IEEE 39-bus system |
|------------------------------------------------------|
| **Parameters** | **Value** |
|----------------|----------|
| Policy Network Size (hidden layers) | [32,32] |
| Number of Directions ($N$) | 16 |
| Top Directions ($N$) | 8 |
| Number of Maximum Iterations ($I$) | 500 |
| Step Size ($\alpha$) | 1 |
| Standard Deviation of Exploration Noise ($\nu$) | 2 |
| Decay Rate ($\epsilon$) | 0.99 |

Based on the above evaluation, we chose the LSTM model to test the parallel scalability of the PARS algorithm, which measured the capacity of effectively using an increasing number of processors. The following two groups of different parallel parameters were investigated:

- number of perturbation directions $N$
- number of CPUs

The influence of increasing the number of policy perturbation directions $N$ on training performance is significant, as shown in Fig. 6(B). Using only eight directions might not be sufficient to archive an acceptable performance, while 16 directions are required to reach optimal training results. Using more directions than 16 will improve the convergence rate but requires more computational resources. The parallel scaling performance with different cores for 16 policy perturbation directions is plotted in Fig. 7(A). It shows that the high-performance computing platform has excellent scalability. The training time is about 5.5 hours for PARS using only 9 cores of one CPU, whereas the training time for DQN is 21 hours in our previous work[4], demonstrating the high efficiency of PARS.

We tested the trained LSTM policy on a set of 120 tasks (fault scenarios) with the combination of 30 different fault locations (bus 1 to bus 30) and four fault duration times (0.02, 0.05, 0.08, and 0.1 s). We also compared the trained ARS-based load shedding control versus the conventional UVLS load shedding scheme. The comparison results show that ARS outperformed the UVLS for all the tasks that required load shedding, as the rewards ARS obtained were always higher than UVLS for those tasks. As a result, either ARS shed less load or UVLS could not recover the system voltage in
Fig. 7. Total time cost using different number of cores. (A) IEEE 39-bus, time cost of 300 iterations, LSTM model, 16 directions; (B) IEEE 300-bus, time cost of 500 iterations, LSTM model, 128 directions.

the required time to meet the standard defined in Fig. 2. Fig. 8 shows the performance comparison between ARS and UVLS for a test task with 0.08 s of fault at bus 3. The total rewards of the ARS and UVLS relay control in this test case were -94.09 and -2367.21, respectively. From Fig. 8(A), it is shown that the voltage with UVLS control (green curve) at bus 4 could not recover within required time to meet the standard (dashed black curve), while the voltage with ARS control (blue curve) could recover to meet the standard. More importantly, Fig. 8(B) shows that the better voltage recovery for ARS is achieved with even less (about 100 MW) total load shedding amount, compared with the UVLS relay control, demonstrating the adaptiveness advantage of ARS over UVLS.

C. Test Case 2: IEEE 300-bus system

Based on the 39-bus system training and testing results, LSTM was chosen to model control policy for the IEEE 300-bus system. The possible load shedding control actions were defined for all buses with dynamic motor loads at zone 1 (46 buses in total), and the amount of load could be shed for each bus at each action time step. This is a continuous variable from 0 (no load shedding) to 0.2 (shedding 20% of the initial total load at the bus). The observations included voltage magnitudes at buses in zone 1 (total 154 buses) as well as the fractions of loads served at the 46 buses where load shedding could be applied; thus the dimension of the input observation was 200. The task set $T$ was defined as 27 different tasks (fault scenarios), which was a combination of 3 fault duration times (0.0, 0.05, and 0.08 s) and 9 candidate fault buses (i.e., 2, 3, 5, 8, 12, 15, 17, 23, 26). At each of the training iterations, 3 fault locations are sampled and combined with 3 potential fault durations to create the rollout tasks. Other parameter settings for the ARS algorithm were the same as Table I, except that the policy network size was increased to $[64, 64]$ and the decay rate was 0.996 (for longer exploration). We trained the LSTM policy with 32, 64, and 128 policy perturbation directions. Fig. 9(A) shows the average rewards with respect to training iterations under different policy perturbation directions and it was clear that the training converged faster and achieved better final rewards with an increased number of perturbation directions. Fig. 7(B) shows the parallel scaling performance on the 300-bus system training with different cores for 128 policy perturbation directions. It also shows that the training of 500 iterations could finish within 7 hours when we fully distributed the computation tasks to 1152 cores, which is about 71 times speedup compared with the same training on 27 cores. In contrast, conducting the same training using three cores takes around 40 days according to the estimation. This means the parallelism implemented in this study speeds up the training by approximately 137 times.

We tested the LSTM policy trained with 128 perturbation directions on a set of 170 different tasks (fault scenarios) with the combination of 34 different fault buses in zone 1 and five fault duration times (0.05, 0.06, 0.07, 0.08, and 0.1 s). We also compared the ARS-based load shedding control versus the conventional UVLS load shedding scheme. To show the comparison results, we calculated the reward differences (i.e., the reward of ARS subtracts that of UVLS) for all the test tasks. A positive value means the ARS method is better for the corresponding test scenario and vice versa. Fig. 9(B) shows the histogram of the rewards differences. As can be seen, ARS outperformed UVLS for 168 out of the 170 tasks (98.82%).

Fig. 10 shows the comparison of ARS and UVLS performance for a test task with 0.1 s of fault at bus 23. The total rewards of the ARS and UVLS relay control in this test case were -823.58 and -21901.80, respectively. Fig. 10(A) shows that the voltage with UVLS control (green curve) at bus 33 could not recover to meet the standard (dashed black curve), while the voltage with ARS control (blue curve) could recover to meet the standard. Further investigation indicated there were five buses that could not recover their voltage with the UVLS control, while the ARS control could bring the voltage at all
buses back. Fig. 10(B) shows that ARS control only shed around 400 MW of load to bring the system voltage back to meet the standard, while the UVLS control shed a total of more than 1000 MW load but still could not recover the system voltage to the level required by the standard.

V. CONCLUSIONS

As described in this paper, a highly scalable DRL algorithm named PARS was developed based on the ARS algorithm and tailored for power system voltage stability control using load shedding. The derivative-free nature and inherent parallelism in the ARS algorithm are fully exploited in PARS.

PARS is developed on the Ray framework and synergistically integrated with the RLGC platform to achieve high scalability and applicability for power system stability control applications. Furthermore, both FNN and LSTM are considered for policy modeling in PARS to better handle high non-linearities in power systems and enhance its generalization capability to unseen scenarios. A small number of hyperparameters makes PARS easy to tune to achieve good performance. Case studies on the IEEE 39-bus demonstrated that LSTM performs better than FNN in terms of computational efficiency and that PARS scales well even for small systems. The high scalability of PARS enables reducing the training time of IEEE 300-bus system from about 40 days to less than 10 hours (i.e., 100X speedup).

Potential future work includes: 1) applying PARS to larger systems and different control applications; 2) combining PARS with meta-learning to exploit past learning experience to further reduce training time for new tasks; and 3) investigating safety- and risk-related issues of DRL-based grid control.

REFERENCES

[1] Australian Energy Market Operator, “Black system South Australia 28 September 2016 - Final Report,” March 2017. [Online]. Available: https://www.aemo.com.au/-/media/Files/Electricity/NEM/Market_Notices_and_Events/Power_System_Incident_Reports/2017/Integrated-Final-Report-SA-Black-System-28-September-2016.pdf
[2] T. Van Cutsem and C. Vournas, Voltage stability of electric power systems. Springer Science & Business Media, 2007.
[3] T. Van Cutsem and C. Vournas, “Emergency voltage stability controls: An overview,” in 2007 IEEE Power Engineering Society General Meeting. IEEE, 2007, pp. 1–10.
[4] Q. Huang, R. Huang, W. Hao, J. Tan, R. Fan, and Z. Huang, “Adaptive power system emergency control using deep reinforcement learning,” IEEE Transactions on Smart Grid, vol. 11, no. 2, pp. 1171–1182, 2020.
[5] A. Stooke and P. Abbeel, “Accelerated methods for deep reinforcement learning,” arXiv preprint arXiv:1803.02811, 2018.
[6] D. Lefebvre, C. Moors, and T. Van Cutsem, “Design of an undervoltage load shedding scheme for the hydro-quebec system,” in 2003 IEEE Power Engineering Society General Meeting (IEEE Cat. No.03CH37491), vol. 4, July 2003, pp. 2030–2036 Vol. 4.
[7] S. Misra, L. Roald, M. Vuffray, and M. Chestnov, “Fast and robust determination of power system emergency control actions,” arXiv preprint arXiv:1707.07105, 2017.
[8] M. Glavic and T. Van Cutsem, “Some reflections on model predictive control of transmission voltages,” in 2006 38th North American Power Symposium, 2006, pp. 625–632.
[9] M. Glavic, D. Novosel, E. Heredia, D. Kosterev, A. Salazar, F. Habibi-Ashrafi, and M. Donnelly, “See it fast to keep calm: Real-time voltage control under stressed conditions,” IEEE Power and Energy Magazine, vol. 10, no. 4, pp. 43–55, July 2012.
[10] A. R. R. Matavalam and V. Ajjarapu, “Pnn-based monitoring and mitigation of delayed voltage recovery using admittances,” IEEE Transactions on Power Systems, vol. 34, no. 6, pp. 4451–4463, 2019.
[11] H. Sun, Q. Guo, J. Qi, V. Ajjarapu, R. Bravo, J. Chow, Z. Li, R. Moghe, E. Nasr-Azadani, U. Tamrakar, G. N. Taranto, R. Tonkoski, G. Valverde, Q. Wu, and G. Yang, “Review of challenges and research opportunities for voltage control in smart grids,” IEEE Transactions on Power Systems, vol. 34, no. 4, pp. 2790–2801, July 2019.
[12] L. GENC, R. Diao, V. Vittal, S. Kolluri, and S. Mandal, “Decision tree-based preventive and corrective control applications for dynamic security enhancement in power systems,” IEEE Transactions on Power Systems, vol. 25, no. 3, pp. 1611–1619, 2010.
[13] Q. Li, Y. Xu, and C. Ren, “A hierarchical data-driven method for event-based load shedding against fault-induced delayed voltage recovery in power systems,” IEEE Transactions on Industrial Informatics, pp. 1–1, 2020.
[14] M. Glavic, R. Fonteneau, and D. Ernst, “Reinforcement learning for electric power system decision and control: Past considerations and perspectives,” IFAC-PapersOnLine, vol. 50, pp. 6918–6927, 2017.
[15] M. Glavic, “(deep) reinforcement learning for electric power system control and related problems: A short review and perspectives,” Annual Reviews in Control, 2019.
[16] J. Zhang, C. Lu, C. Fang, X. Ling, and Y. Zhang, “Load shedding scheme with deep reinforcement learning to improve short-term voltage stability,” in 2018 IEEE Innovative Smart Grid Technologies-Asia (ISGT Asia). IEEE, 2018, pp. 13–18.
[17] L. Espeholt, R. Marinier, P. Stanecky, K. Wang, and M. Michalski, “Seed r1: Scalable and efficient deep-rl with accelerated central inference,” arXiv preprint arXiv:1910.06591, 2019.
[18] H. Mania, A. Guy, and B. Recht, “Simple random search of static linear policies is competitive for reinforcement learning,” in Advances in Neural Information Processing Systems, 2018, pp. 1800–1809.
[19] T. Taylor, “Concepts of undervoltage load shedding for voltage stability,” IEEE Transactions on Power Delivery, vol. 7, no. 2, pp. 480–488, Apr. 1992, conference Name: IEEE Transactions on Power Delivery.
[20] H. Bai and V. Ajjarapu, “A novel online load shedding strategy for mitigating fault-induced delayed voltage recovery,” IEEE Transactions on Power Systems, vol. 26, no. 1, pp. 294–304, 2011.
[21] R. S. Sutton and A. G. Barto, Introduction to reinforcement learning. MIT press Cambridge, 1998, vol. 135.
[22] V. François-Lavet, P. Henderson, R. Islam, M. G. Bellemare, and J. Pineau, “An introduction to deep reinforcement learning,” arXiv preprint arXiv:1811.12560, 2018.
[23] T. Salimans, J. Ho, X. Chen, S. Sidor, and I. Sutskever, “Evolution strategies as a scalable alternative to reinforcement learning,” 2017.
[24] R. Huang, S. Jin, Y. Chen, R. Diao, B. Palmer, Q. Huang, and Z. Huang, “Faster than real-time dynamic simulation for large-size power system with detailed dynamic models using high-performance computing platform,” in 2017 IEEE Power & Energy Society General Meeting. IEEE, 2017, pp. 1–5.
[25] PJM Transmission Planning Department, “Exelon transmission planning criteria,” 2009.
[26] R. S. Sutton and A. G. Barto, Introduction to reinforcement learning. MIT press Cambridge, 1998, vol. 135.
[27] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” Neural Comput., vol. 9, no. 8, p. 17351780, Nov. 1997. [Online]. Available: https://doi.org/10.1162/neco.1997.9.8.1735
[28] P. Moritz, R. Nishihara, S. Wang, A. Tumanov, R. Liaw, E. Liang, M. Elbol, Z. Yang, W. Paul, M. I. Jordan, and I. Stoica, “Ray: A distributed framework for emerging ai applications,” 2017.
[29] Q. Huang, W. Hao, and R. Huang, “RLGC: An open-source platform for developing and benchmarking reinforcement learning for grid control,” https://github.com/RLGC-Project/RLGC, accessed: 2018-12-10.