Accelerated Derivative-Free Deep Reinforcement Learning for Large-Scale Grid Emergency Voltage Control

Renke Huang, Yujiao Chen, Tianzhixi Yin, Xinya Li, Ang Li, *Member, IEEE*, Jie Tan, Wenhao Yu, Yuan Liu, *Member, IEEE*, and Qiuhua 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 novel derivative-free DRL algorithm named PARS was developed and tailored for power system voltage stability control via load shedding. 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 including model-predictive control (MPC) and proximal policy optimization(PPO) methods, PARS shows better computational efficiency (faster convergence), more robustness in learning, 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 instability 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 counteracting voltage instability, in particular short-term voltage instability [2]–[4]. On the one hand, short-term voltage stability events are fast and usually last 5 to 30 seconds. Thus, it requires fast solutions and fast actions (e.g., less than half a second for each control interval [5]) to control and mitigate them in real-time operation. On the other hand, effectively determining load shedding time, location and amount for large power system is a complex, dynamic, and sequential decision-making problem [4]. 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 a novel derivative-free, parallel augment random search (PARS) algorithm for large-scale power systems to make load shedding for emergency voltage control fast, adaptive, and scalable.

To date, it takes extensive time (ranging from tens of hours to days) to train a practically good control policy for even small-scale power systems using existing state-of-the-art DRL algorithms [4]. Accelerated training with good performance not only makes DRL more practical for real-world large-scale power grid control applications, but also brings significant benefits, including: 1) shorter experiment turnaround time facilitating better control design [6]; 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,

Manuscript received June 28, 2020; revised November 12, 2020 and April 14, 2021; accepted June 20, 2021. Date of publication July 7, 2021; date of current version December 23, 2021. This work was supported by U.S. Department of Energy (DOE) ARPA-E OPEN 2018 Program. The Pacific Northwest National Laboratory is operated by Battelle for the U.S. DOE under Grant DE-AC05-76RL01830. Paper no. TPWRS-01070-2020. *(Corresponding author: Qiuhua Huang.)*

Renke Huang, Yujiao Chen, Tianzhixi Yin, Xinya Li, Ang Li, Yuan Liu, and Qiuhua Huang are with PNNL, Richland, WA 99354 USA (e-mail: renke.huang@pnnl.gov; chenyujiao2020@gmail.com; tianzhixi.yin@pnnl.gov; xinya.li@pnnl.gov; ang.li@pnnl.gov; yuan.liu@pnnl.gov; qiuhua.huang@pnnl.gov).

Jie Tan and Wenhao Yu are with Google Brain, Google Inc, Mountain View, CA 94043 USA (e-mail: jietan@google.com; magicmelon@google.com).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TPWRS.2021.3095179.

Digital Object Identifier 10.1109/TPWRS.2021.3095179

model-based, and learning-based (including data-driven) approaches.

1) Rule-based approaches: Most existing load shedding control designs deployed in the power industry are rule-based, which could provide the load shedding control actions in a very short time since the rules are determined based on off-line studies and do not change online for different operation conditions or faults. A simple scheme relies on simple rules like "if voltage drops below some threshold V_{th} for some duration τ, shed some power ΔP" [3]. In addition, the settings of the rules tend to be conservative. While some enhancements to simple rule-based methods were proposed [7], the main issues of such methods are lack of adaptiveness and optimality. The main reason is that rule-based methods do not consider the spatial and temporal sensitivity and correlation of load shedding actions with respect to the effectiveness of recovering the system voltages, which is critical for achieving effective and adaptive emergency controls. This spatial and temporal sensitivity and correlation of load shedding actions vary over time for different operation conditions and contingencies, thus it cannot be captured by fixed rules.

2) Model-based approaches: Security-constrained alternating current optimal power flow (AC-OPF) [8] was proposed for grid emergency control. Another widely used model-based approach is model-predictive control (MPC) [9], [10]. MPC could provide optimal or near-optimal solutions for dynamic, sequential load shedding optimization problems. However, model-based methods suffer from poor scalability and long solution time issues for large-scale grid control problems, due to the computational complexity. Consequently, they can not meet the short response time requirement of grid emergency control (usually less than half a second [5]). In addition, since they rely on a system model, these methods are susceptible to model inaccuracies [4].

3) Measurement-based approaches: 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 [11]–[13]. The measurement-based approach can estimate the "sensitivity" of local load shedding actions to determine the appropriate load shedding amount for each local load. In [11] an online load shedding strategy by estimating the motor kinetic energy using PMU voltage and current measurements is proposed to determine the load shedding amount. In [12], the load shedding actions that ensure voltage recovery within a pre-specified time are determined by an index calculated based on active and reactive power measurements. The measurement-based method could meet the response time requirement for emergency controls, since the proposed algorithms for estimating the sensitivity indices typically have a lower computation complexity when compared with model-based methods. However, without high-level coordination that would require some model-based or learning-based approaches, these methods may not be adequate to mitigate emergencies on large-size power grids.

4) Learning-based approaches: Learning-based (or data-driven) methods gained much attention and interest for grid control in both academia and industry in the past decade. The basic idea of learning-based methods is to learn a (usually non-linear) mapping between the system operating conditions and control actions from data (mostly collected from simulation), and the mapping is encapsulated in machine learning models such as a neural network. During the online application, the trained model can infer control decisions within a very short time based on the actual system conditions. A decision-tree-based approach was proposed for preventive and corrective control [14]. A hierarchical, extreme learning machine-based method for load shedding against fault-induced delayed voltage recovery (FIDVR) events was developed in [15]. An important learning-based approach for controlling dynamic systems is reinforcement learning (RL). There is a significant number of previous efforts in utilizing conventional RL methods such as Q-learning for many different power system control applications and areas [16], [17]. 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. These limitations are largely overcame by recent advancement of DRL algorithms, which have 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 [18]. DDPG has also been applied for power grid emergency frequency control in [19], [20].

It should be noted that many state-of-the-art DRL algorithms such as DQN, DDPG, and PPO used in [4], [18]–[20] are notoriously known for difficulty in scaling up the solutions, and time-consuming in training and hyper-parameter tuning for large-size power grid applications. We have identified the following challenges when applying and testing the existing value-function-based and policy-gradient-based DRL methods for this emergency control problem on a large-scale power grid:

1) The action space exploration mechanism of the value-function and policy-gradient-based DRL methods is not effective enough for large-scale grid control problems.
2) The large-scale grid emergency voltage problem exhibits non-smoothness in the environment and the reward function definition. Thus, the gradient computation approach in value-function and policy-gradient-based DRL methods will suffer the exploding gradient issue.
3) The value-function-based and policy-gradient-based DRL methods are challenging to have a fully parallel implementation because there are complex dependencies between different components of these algorithms [21].
4) Many value-function and policy-gradient-based DRL algorithms are highly sensitive to the values of hyperparameters and random seeds [22], [23]. Considerable manual hyper-parameter tuning is required to make these DRL methods work well.
5) The combination of challenges 3 and 4 leads to very long training and hyper-parameter tuning time to achieve acceptable performance with the existing DRL methods

on the large-scale power grid emergency voltage control problem.

C. Contributions

Different from our previous work [4] that focused on adapting and applying DQN algorithm for two discrete grid emergency control schemes on small-size power grids, a novel, much more robust and scalable, yet easy to tune PARS algorithm is developed in this paper to tackle both the scalability and training efficiency issues identified above. The main contributions of this paper include:

1) A novel derivative-free PARS algorithm is developed and integrated with the long short-term memory (LSTM) neural network architecture for the grid emergency voltage control problem. LSTM helps mitigate the non-Markovian issue of the problem. The derivative-free optimization of PARS enables the RL agent to support much larger learning rates and has much fewer hyper-parameters to tune compared with existing DRL methods. The combination of both makes our solution well-suited for solving such high-dimensional, and highly nonlinear power system control problems.
2) The proposed PARS algorithm performs structured and much more effective parameter-space exploration during large-scale RL training when compared with existing state-of-the-art DRL algorithms that adopt action-space exploration.
3) The proposed PARS algorithm uses a simultaneous perturbation stochastic approximation approach to estimate the gradient and thus avoid the exploding gradient issue associated with the non-smoothness of the defined power grid voltage control problem, while other back-propagation-based DRL algorithms suffer from it.
4) The proposed two-level parallelism architecture makes the PARS-base learning and grid dynamic simulation highly scalable. The communication overhead of implementing PARS in a distributed setting is much lower than other DRL methods. Over 90% of the computation in the PARS can be easily parallelized and thus the learning speed almost scales linearly with the number of CPU cores, while other DRL methods experience a saturation or bottleneck of acceleration with smaller number of cores.

D. Organization

The rest of the paper is organized as follows: Section II introduces DRL and the problem formulation. Section III presents the PARS algorithm and its enhancements and advantages for large-size grid control problem. 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 [24]. 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 [25] 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 optimal control problem and its conversion to a Markov decision process (MDP) formulation can be found in the recent work of the authors [4]. Note that the constrained optimal control problem for load shedding control formulated in [4] includes the equality constraints of dynamic differential equations of the grid, as well as inequality constraints of the dynamic states of the grid. Thus, the size of the optimization problem becomes very large for large power grids and it is computationally intensive to solve this problem with model-based MPC methods [4]. As this paper focuses on an accelerated 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 $(\mathcal{S}, \mathcal{A}, \mathcal{P}, r)$ [26]. The state space $\mathcal{S}$ and action space $\mathcal{A}$ could be continuous or discrete. In this paper, both of them are continuous. The environment transition function $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \longrightarrow \mathbb{R}$ is the probability density of the next state $s_{t+1} \in \mathcal{S}$ given the current state $s_t \in \mathcal{S}$ and action $a_t \in \mathcal{A}$. At each interaction step, the environment returns a reward $r : \mathcal{S} \times \mathcal{A} \longrightarrow \mathbb{R}$. The standard RL objective is the expected sum of discounted rewards. The goal of an agent is to learn a policy $\pi_\theta(s_t, a_t)$ that maximizes the objective.

DRL is a combination of RL and deep-learning technologies. The capabilities of high dimensional feature extraction and nonlinear 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.

There are three classes of RL algorithms, i.e., model-based RL, model-free RL and derivative-free RL [27]. Fig. 1 shows comparisons of them in terms of computational scalability. Computational scalability is mainly related to parallelization and speeding up the training process. In addition to scalability, main factors for determining a proper RL algorithm for one particular application include sampling efficiency and hyperparameter tuning. Sampling efficiency measures how much data need to be collected to train the RL algorithm. Regarding hyperparamter tuning, the fewer sensitive hyperparameters one RL

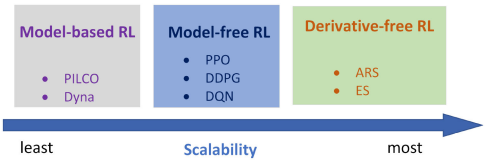

Fig. 1. Comparison of different classes of RL algorithms.

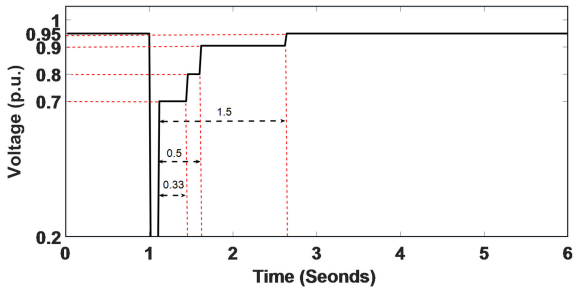

Fig. 2. Transient voltage recovery criterion for transmission system [31].

algorithm has, the easier (and less time) it takes to tune them to achieve good performance. Generally, it is very rare that one RL algorithm has outstanding performance in all these aspects, thus the key is to select one that can meet the requirements of grid emergency control and has good potential to overcome the bottlenecks in training time and scalability.

For model-based RL, the planning stage has not been able to scale reliably to high dimensions, leading to the poor scalability of model-based approaches [27]. For model-free RL, when neural networks are used as function approximation for the policy, 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. Recently, there are several derivative-free methods such as ARS [28] and natural evolution strategies (ES) [29] that have been developed as competitive and highly scalable alternatives to other gradient-based DRL algorithms.

Admittedly, ARS is not the most sampling (or data) efficient DRL algorithm. However, data efficiency is not the bottleneck for power grid emergency control problems because power grid simulators can be used to generate data efficiently on the fly, in parallel, and faster than real-time [30]. On the other hand, the training speed becomes a bottleneck. Many DRL algorithms need to train for days or even weeks before good policies can be learned. Such slow training time makes these methods impractical for emergency control of large-scale power grids, where both the simulation and the control policy to be learned are considerably more complex. Given the current trend that computation becomes cheaper and more computing resources are readily available, we purposefully choose a scalable learning algorithm that can take advantage of this trend. It should be noted that the small number of sensitive hyper-parameters 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 enhanced and accelerated it to achieve the object of designing a fast, adaptive, and scalable DRL-based load shedding scheme in this paper, and thus overcame the bottlenecks in training speed and scalability. We have successfully applied our method on the IEEE 300-bus system and reduced the training time by 136 times.

C. MDP Formulation for Load Shedding

1) State: The observed system variables $\mathbf{O}_t$ at time t include voltage magnitudes at monitored buses (denoted as V_t) 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., $s_t = (\mathbf{O}_{t-N_r-1}, \ldots, \mathbf{O}_t)$.

2) 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.

3) 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].

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 [31] 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:

$$r_t = \begin{cases} -M, \text{ if } V_i(t) < 0.95, \quad t > T_{pf} + 4 \\ c_1 \sum_i \Delta V_i(t) - c_2 \sum_j \Delta P_j(t) - c_3 u_{ivld}, \text{ otherwise} \end{cases} \tag{1}$$

$$\Delta V_i(t) = \begin{cases} min\left\{V_i(t) - 0.7, 0\right\}, \text{ if } T_{pf} < t < T_{pf} + 0.33 \\ min\left\{V_i(t) - 0.8, 0\right\}, \text{ if } T_{pf} + 0.33 < t < T_{pf} + 0.5 \\ min\left\{V_i(t) - 0.9, 0\right\}, \text{ if } T_{pf} + 0.5 < t < T_{pf} + 1.5 \\ min\left\{V_i(t) - 0.95, 0\right\}, \text{ if } T_{pf} + 1.5 < t \end{cases}$$

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, or 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 if any bus voltage is below 0.95 p.u. 4 s after the fault is cleared, the reward is set to a large negative number $-M$, which is -1000 for the IEEE 39-bus system and $-10\,000$ for the IEEE 300-bus system test cases in this paper.

Note that the objective of the RL algorithm is to maximize the total accumulated rewards over a specific time period T_{max}, i.e., $max \sum_{t=0}^{T_{max}} \gamma^t r_t$, where γ is a discount factor. We designed the reward function defined in Eqn. (1) in such a way that maximizing the total accumulated rewards will result in minimizing the total load shedding amount and the voltage violations at all measured buses with reference to the voltage performance envelop curve in Fig. 2. In other words, the optimality of the power grid load shedding control problem is achieved by shedding as less load as possible to recover voltage buses to meet the voltage performance requirements.

III. ARS Algorithm and its Enhancements

A. ARS Algorithm

In the context of RL, given the reward function r and the policy $\pi(s|\theta)$, the goal of the ARS algorithm is to find the optimal policy weights θ that maximizes the expected total discounted reward $E\{\sum_{t=0}^{T} \gamma^t r_t(s_t, \pi(s_t|\theta))\}$. Different from existing model-free DRL algorithms that use action-space exploration, ARS performs policy parameter-space exploration and estimates the gradient of the returns using simultaneous perturbation stochastic approximation, thus back-propagation is not needed. Compared to existing DRL algorithms (i.e., TRPO, DDPG, PPO, A2C, and SAC), Mania *et al.* [28] 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., α, υ, N, b, and m in **Algorithm** 1, which makes it much easier for end users to achieve satisfactory control performance without extensive hyperparameters tuning.

As shown in the pseudo code of **Algorithm** 1, the ARS algorithm consists of two repeated phases: first, the policy parameter is randomly perturbed by noises derived from a Gaussian distribution, and the associated actions are executed and evaluated based on their reward values for an entire episode (step 5-step 8); second, the policy parameter is updated by an estimated stochastic gradient (steps 10 and 11), which comes from the following derivation: assuming our objective is to optimize θ over a distribution $p_\psi(\theta)$ to maximize the expected reward $\mathbb{E}_{\theta \sim p_\psi(\theta)} r(\theta)$, when the parameter distribution $p_\psi(\theta)$ follows a Guassian distribution, the expected reward can be directly written as $\mathbb{E}_{\theta \sim p_\psi(\theta)} r(\theta) = \mathbb{E}_{\delta \sim N(0,I)} r(\theta + \upsilon\delta)$. With the objective defined in terms of θ, the gradient can be calculated as follows:

$$\nabla \mathbb{E}_{\delta \sim N(0,I)} r(\theta + \upsilon\delta) = \frac{1}{\delta} \mathbb{E}_{\delta \sim N(0,I)} \{r(\theta + \upsilon\delta)\delta\} \qquad (2)$$

The expectation term in (2) can be achieved through sampling, as shown by step 4 in the algorithm. Note that the steps 4 - 8 of the algorithm have inherent parallelism as the evaluation of each perturbation direction is independent. The challenge of scaling-up is that they are integrated with complex, and computationally expensive power grid dynamic simulation. To scale up the ARS algorithm for large-scale grid control problems and reduce the training time, we proposed a novel nest parallelism scheme and implemented it on a high-performance computing platform. We also introduced a decay for step size and the perturbation noise. Details are discussed in Subsection III-B.

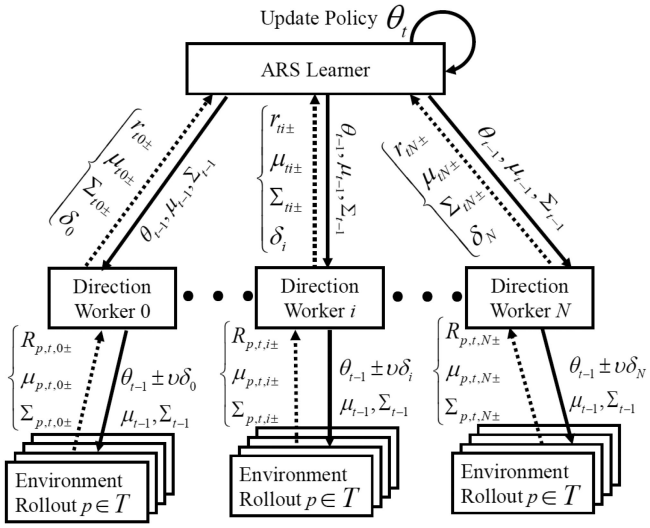

Fig. 3. A nest parallel scheme for accelerating the ARS algorithm.

The original ARS algorithm was proposed for linear control policies with the policy θ represented by a matrix [28]. 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 this shortcoming, we enhanced ARS by modeling policies with neural networks, namely the FNN and the LSTM network [32]. 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 [33], which supports task parallelism (via Ray remote functions) and actor-based computation (via Ray remote classes). We developed a hierarchical distributed-computing architecture, as illustrated in Fig. 3, to fully utilize the parallelism of our problem formulation. In our formulation, both the learning algorithm and the simulation are parallelized. The first layer of our hierarchical architecture is for

Algorithm 1: Modified ARS.

1: **Hyperparameters:** Step size α, number of policy perturbation directions per iteration N, standard deviation of the exploration noise υ, number of top-performing perturbed directions selected for updating weights b, number of rollouts per perturbation direction m. Decay rate ε.
2: **Initialize:** Policy weights θ_0 with small random numbers; the running mean of observation states $\mu_0 = \mathbf{0} \in \mathbb{R}^n$ and the running standard deviation of observation states $\Sigma_0 = \mathbf{I}_n \in \mathbb{R}^n$, where n is the dimension of observation states, the total iteration number H.
3: **for** iteration $t = 1, \ldots, H$ **do**
4: sample N random directions $\delta_1, \ldots, \delta_N$ with the same dimension as policy weights θ
5: **for** each $\delta_i (i \in [1, \ldots, N])$ **do**
6: add $\pm$ perturbations to policy weights: $\theta_{ti+} = \theta_{t-1} + \upsilon\delta_i$ and $\theta_{ti-} = \theta_{t-1} - \upsilon\delta_i$
7: **do** total 2 m rollouts (episodes) denoted by $R_{p\in T}(\cdot)$ 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_{ti+} and r_{ti-}

$$\begin{cases} r_{ti+} = \frac{1}{m} R_{p\in T}(\theta_{ti+}, \mu_{t-1}, \Sigma_{t-1}) \\ r_{ti-} = \frac{1}{m} R_{p\in T}(\theta_{ti-}, \mu_{t-1}, \Sigma_{t-1}) \end{cases} \tag{3}$$

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 π_{θ_t} to obtain the action $a_{t,k}$, which is applied to the environement and new states $s_{t,k+1}$ is returned, as shown in (4). The running mean μ_t and standard deviation Σ_t are updated with $s_{t,k+1}$

$$\begin{cases} s_{t,k} = (s_{t,k} - \mu_{t-1})/\Sigma_{t-1} \\ a_{t,k} = \pi_{\theta_t}(s_{t,k}) \\ s_{t,k+1} \longleftarrow \mathcal{P}(s_{t,k}, a_{t,k}) \end{cases} \tag{4}$$

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

$$\theta_{t+1} = \theta_t + \frac{\alpha}{b\sigma_b} \sum_{i=1}^{b} (r_{ti+} - r_{ti-})\delta_i \tag{5}$$

12: Step size α and standard deviation of the exploration noise υ decay with rate ε: $\alpha = \varepsilon\alpha$, $\upsilon = \varepsilon\upsilon$
13: **end for**
14: **return** θ

parallelizing the learning (step 5 in **Algorithm** 1) and the second layer is for parallelizing the environment simulation (step 7 in **Algorithm** 1). Although it is possible to collapse this two-layer hierarchy into one level, analogue to reshaping a 2D matrix into one vector, we deliberately choose this hierarchical structure for the following three reasons. 1) Conceptual simplicity: It is a cleaner and easy-to-maintain design to have different layers for different components (learning vs. simulation). 2) Support of different strategies for sampling tasks: it allows deliberately sampling of tasks of varied levels of difficulty in learning for each iteration in training. 3) Modularity and extensibility to support full parallelism: currently in the algorithm we rollout the environment multiple times using the same perturbed policy to account for the stochasticity in the simulated environment. We also plan to further parallelize the single rollout simulation with multiple CPU cores when we apply our method on larger power grids (e.g., more than 2000 buses) in the future [30]. It is much easier to implement such extensions with the current hierarchical design: we can simply add more layers without major changes to the existing code.

As shown in Fig. 3, the ARS learner is an actor at the top to delegate tasks and collect returned information, and controls the update of policy weights θ. 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 RLGC [34], which is an open source tool for developing and benchmarking RL algorithms for grid control and supports task-level parallelism.

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 a simultaneous perturbation stochastic approximation approach (5) 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. 4(a) 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

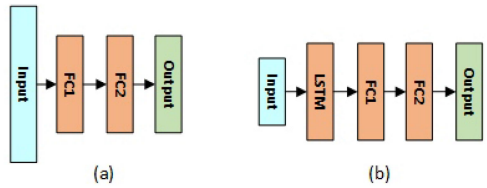

Fig. 4. Neural network architectures integrated with ARS: (a) FNN; (b) LSTM+FNN.

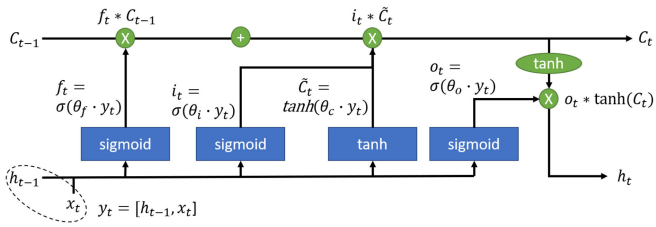

Fig. 5. LSTM network structure.

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). Note that LSTM is notoriously difficult to train for RL problems with non-smoothness features by using other value-function-based or policy-gradient-based DRL approaches, as a result of the exploding gradient issue associated with the back-propagation-based gradient computation [29]. Thus, we apply a simultaneous perturbation stochastic approximation approach (5) in Algorithm 1 to train the LSTM, which does not need to back-propagate gradients, and makes the learning much more robust and stable.

D. Comparison With Existing DRL Methods

Here we summarize the distinguished advantages and features of the proposed PARS algorithm compared with other DRL algorithms as follows.

1) PARS performs structured and much more effective exploration for large-scale RL problem. Exploration is critical for DRL to find optimal policies. Most DRL algorithms adopt action-space exploration, for example, by adding Gaussian noise to the output action of the neural network policy. This exploration can cause a random jitter on spot, search only locally, and is not an effective exploration strategy for large-scale RL problems and long-horizon decision making. In contrast, the proposed PARS method uses parameter-space exploration. Instead of perturbing the actions randomly, it perturbs the policies. Parameter-space exploration was shown to lead to more effective exploration, a richer set of behaviors, and more successful training [35].
2) PARS does not need back-propagation to compute the gradients of the environment or policy. For the RL formulation of power grid emergency load shedding problem, the environment and the reward function definition are not smooth, since the load shedding actions at specific time steps will cause abrupt voltage changes in the environment, as well as the invalid action penalty and the failure penalty of the reward function are also discrete. Such non-smoothness of the RL problem formulation will lead to the exploding gradient issue for back-propagation-based gradient computation method used in both value-based (such as DQN) or policy-based (such as PPO) RL algorithms. In contrast, the proposed ES-based ARS algorithm does not explicitly calculating an analytical gradient, it uses a simultaneous perturbation stochastic approximation approach to estimate the gradient and thus avoid the exploding gradient issue associated with the non-smoothness of the defined power grid load shedding RL problem.
3) PARS is highly scalable. The communication overhead of implementing PARS in a distributed setting is lower than other policy-gradients-based and value-function-based RL methods, as the only information that needs to be communicated across processes are the scalar return and the random seed that was used to generate the perturbations, rather than a full gradient. Over 90% of the computation in the PARS can be easily parallelized. For this reason, the learning speed almost scales linearly with the number of CPU cores. This feature is particularly important for applications that require huge amounts of environment simulation computation, such as reinforcement learning for large-scale power grid emergency control, where the dynamics simulation of large-size power grid takes more than 90% of the computations time. In our next section, we used over 500 CPUs for training, achieving a speed up of 136 times compared to its series implementation. Although researchers have devoted significant efforts to parallelize other DRL algorithms, such as the state-of-the-art PPO algorithm that we compared with in the paper in the next section, it is much difficult to have a fully parallel implementation because there are complex dependencies between different components of these algorithms [21], and thus only a small portion of the algorithms can run in parallel. Consequently, as shown in Section IV-D, with the same amount of computational resources, our PARS algorithm completes the training five times faster than the state-of-the-art parallel PPO method.

4) PARS is a more robust learning algorithm. One of the main shortcomings of many DRL algorithms is their sensitivity to the choice of hyperparameters and random seeds [22], [23]. Considerably manual tuning is required to make these methods work. In contrast, PARS has only five hyperparameters, as discussed in Section III-A. Even better, PARS is less sensitive to hyperparameters and random seeds. As shown in Section IV-D, we kept the hyperparameters fixed, and PARS learned consistently high-performing policies for different random seeds. Contrarily, the parallel PPO algorithm had large variance in the training performance, and failed to learn satisfactory control policies more frequently.

One weakness of our PARS method is that it requires at least a modest parallel computing environment (e.g., with 10+ CPU cores) to perform well. However, this is not a bottleneck as such a computing setup is quite common now, even in laptops.

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 and trained NN models are publicly available in [34].

A. Baseline Methods

1) UVLS: it is one widely used rule-based method in the industry nowadays. The following 3-stage settings are used in this paper: 1) $V_{th} = 0.70\,pu, \tau = 0.33\,s, \Delta P = 20\% P_{init}$; 2) $V_{th} = 0.80\,pu, \tau = 0.5\,s, \Delta P = 20\% P_{init}$; 3) $V_{th} = 0.90\,pu, \tau = 1.5\,s, \Delta P = 20\% P_{init}$; where P_{init} denotes the pre-fault total load at a load bus.
2) MPC: it is a widely used analytic-based method for dynamic, sequential control problems. A trajectory-sensitivity based MPC [10] is chosen for comparison.
3) Parallel PPO: It is a state-of-the-art DRL method and has been used to solve large-scale problems such as OpenAI Five. We chose the high-performance parallel PPO implementation in RLlib [21] as a baseline that represents existing model-free RL methods.

B. 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 optimality 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 methods is presented in the following subsections.

TABLE I
Hyperparameters for Training IEEE 39-Bus System and 300-Bus System

Parameters	39-Bus	300-Bus
Policy Network Size (Hidden Layers)	[32,32]	[64,64]
Number of Directions (N)	16	(32,64,128)
Top Directions (b)	8	(16,32,64)
Step Size (α)	1	1
Std. Dev. of Exploration Noise (υ)	2	2
Decay Rate (ε)	0.99	0.996

C. 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 observations 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 continues 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 PARS algorithm interacted with the system with and without FIDVR conditions. Other parameter settings for the PARS 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

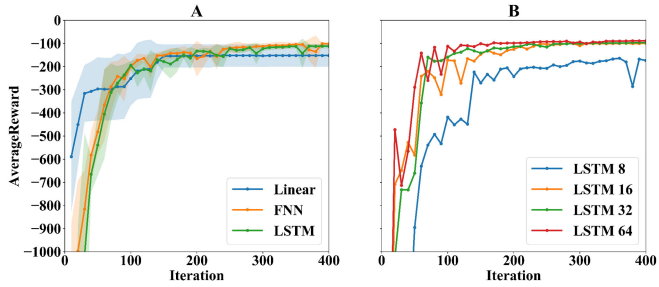

Fig. 6. Convergence curves of training using (A) linear, FNN, and LSTM with 16 directions. The training curves were averaged over five random seeds and the shaded region shows the standard deviations; (B) different numbers of directions for LSTM.

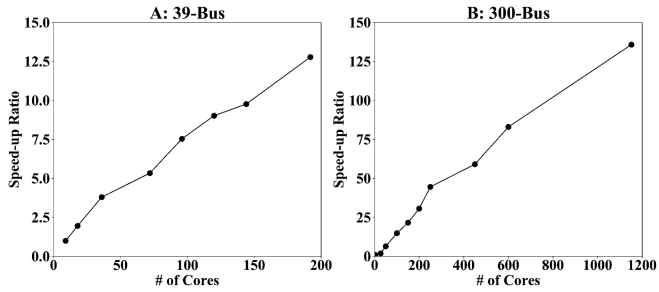

Fig. 7. Speed-up ratio using different number of cores. (A) IEEE 39-bus, 300 iterations, LSTM model, 16 directions, 9-core time cost as base case; (B) IEEE 300-bus, 500 iterations, LSTM model, 128 directions, 3-core time cost as base case.

the computation tasks. As a result, eight computational nodes with maximum available cores of 192 were used for training. Fig. 6(A) shows the reward achieved by PARS 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).

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

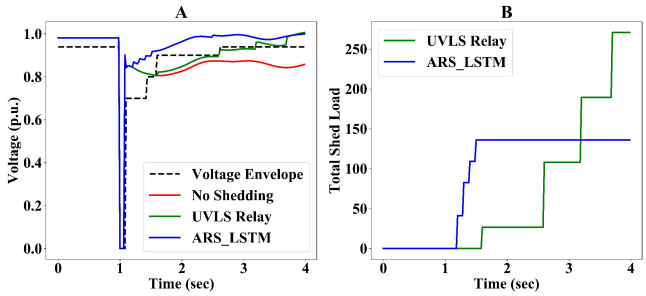

Fig. 8. Testing results of trained LSTM 16 model. (A) Voltage of bus 4. Dash line denotes the performance requirement for voltage recovery. (B) Total load shedding amount.

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 4 fault durations (0.02, 0.05, 0.08, and 0.1 s). We also compared the trained PARS-based load shedding control versus the conventional UVLS load shedding scheme. The comparison results show that PARS outperformed the UVLS for all the tasks that required load shedding, as the rewards PARS obtained were always higher than UVLS for those tasks. As a result, either PARS shed less load or UVLS could not recover the system voltage within the required time period to meet the standard defined in Fig. 2. Fig. 8 shows the performance comparison between PARS and UVLS for a test task with 0.08 s of fault at bus 3. The total rewards of the PARS 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 PARS control (blue curve) could recover to meet the standard. More importantly, Fig. 8(B) shows that the better voltage recovery for PARS is achieved with even less (about 100 MW) total load shedding amount, compared with the UVLS relay control, demonstrating the adaptiveness advantage of PARS over UVLS.

D. 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). The amount of load could be shed for each bus at each action time step 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)

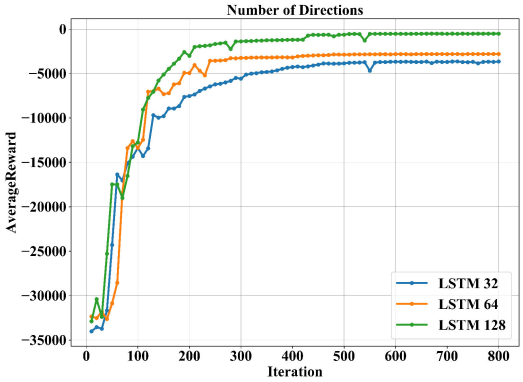

Fig. 9. Convergence curves of training using different numbers of directions for the 300-bus system.

TABLE II
HYPERPARAMETERS OF THE PARALLEL PPO ALGORITHM DIFFERENT FROM DEFAULT VALUES

Parameters	300-Bus
Number of rollout worker actors	500
Learning rate	5e-4
Entropy coeff. (Regularizer)	1e-4
Number of hidden layers for fully connected net	[64,64]
Clip parameter	0.3
Use KL loss	True
VF clip parameter	200

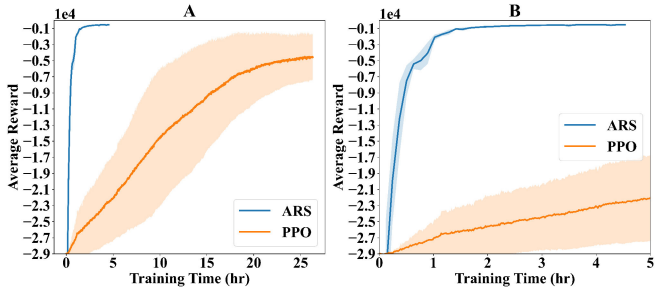

Fig. 10. Convergence curves of training by time (hr) using PARS and PPO for the 300-bus system for (A) 25 hours (B) 5 hours. The training curves were averaged over five random seeds and the shaded region shows the standard deviation.

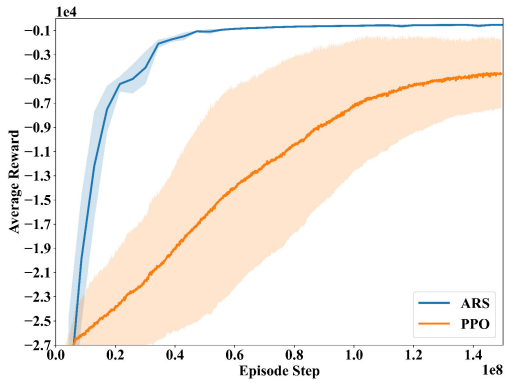

Fig. 11. Convergence curves of training by episode steps using PARS and PPO for the 300-bus system. The training curves were averaged over five random seeds and the shaded region shows the standard deviation.

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. The hyperparameters of the PARS algorithm for the training are shown in Table I. We trained the LSTM policy with 32, 64, and 128 policy perturbation directions. Fig. 9 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. The proposed PARS algorithm gains a speed-up of 136 times when scaling the training from 3 cores to 1152 cores. More importantly, the speed-up curve is almost linear and shows no saturation for up to 1152 cores, suggesting potential for further acceleration when necessary.

We ran the training with PARS and parallel PPO using fixed hyper-parameters but different random seeds for five times. Table II listed hyperparameters of PPO that are different from the default values provided in the RLlib implementation. Both trainings of PARS and PPO use the same computational resources of 500 CPU cores. Fig. 10 shows the learning curves for both PARS and PPO, with mean and variance, with respect to the wall-clock training time. Fig. 10 shows that with the same amount of computational resources, our PARS algorithm runs five times faster than the parallel PPO algorithm. Furthermore, our PARS algorithm learns consistently high-performing policies for all the five training cases (the narrow shaded area indicates small variance). In contrast, the performance of PPO has a large variance for the five training cases (the orange shaded region is much wider than that of PARS), and the converged average reward of PPO is lower than that of PARS. Fig. 11 shows the learning curve for both PARS and PPO, with mean and variance, with respect to the environment simulation steps. Fig. 11 clearly shows that PARS outperforms the PPO in terms of convergence rate in learning. With the same amount of generated data from simulation environment (the same episode steps), PARS achieved notably higher converged average rewards, as well as better successful rate of training (higher converged average rewards and much smaller rewards deviations). The results demonstrate that for the studied grid voltage emergency control problems, our proposed PARS method achieves much better performance than the state-of-the-art parallel PPO method.

After training, we tested the LSTM policy trained with 128 perturbation directions on a set of 170 different tasks (fault scenarios) which correspond to combinations of 34 different fault buses in zone 1 and 5 fault duration times (0.05, 0.06, 0.07, 0.08, and 0.1 s). We also compared the PARS-based load shedding control with the conventional UVLS scheme, the best trained PPO policy, and the solutions obtained by the MPC method.

To show the comparison results, we calculated the reward differences (i.e., the reward of PARS subtracts that of a baseline

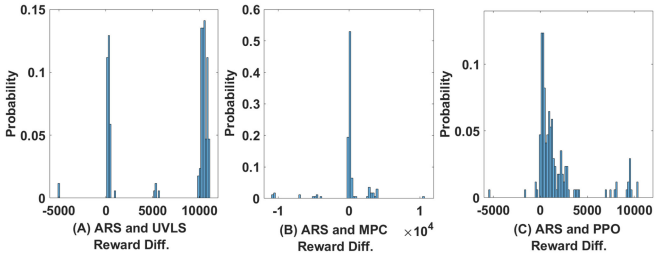

Fig. 12. Histogram of the reward differences: (A) PARS and UVLS; (B) PARS and MPC; (C) PARS and parallel PPO.

TABLE III
COMPARISON OF AVERAGE COMPUTATION TIME FOR PARS AND MPC

PARS	MPC
0.72 seconds	63.31 seconds

method) for all the test cases. A positive value means our PARS method performs better for the corresponding test scenario with respect to the optimality defined by Equ. (1) and vice versa. Fig. 12 shows the histogram of the rewards differences. As can be seen, PARS outperforms both UVLS and PPO since most of the reward differences are positive when compared with UVLS (98.82%) and PPO (95.29%). However, it should be noted that PPO actually has a much better performance when compared with ULVS. Fig. 12(A) shows that for more than 50% test tasks, PARS achieves a reward difference more than 10 000 compared with UVLS, which indicates that for these tasks, ULVS fails to recover the voltages back to meet the requirements and thus get a large "failure" penalty. In contrast, there are only 3 out of 170 tasks where PPO fails, and Fig. 12(C) shows that for about 85% of the test tasks, the reward difference between PARS and PPO are less than 5000. Fig. 12(B) shows that PARS and MPC achieve comparable results. Table III shows the average computation time of the PARS and MPC methods. The computation time for UVLS relays is not included as it is either instantaneous or a predefined delay. It is obvious that PARS requires much shorter solution time than MPC, because performing NN forward pass to infer actions in the PARS approach is much more efficient compared to a solving a complex optimization problem with the MPC method. With 0.72 s action time during a 8-second FIDVR event, the PARS method can meet the real-time operation requirements and allows grid operators to verify the control actions when necessary.

Fig. 13 shows the comparison of PARS, UVLS, PPO, and MPC performance for a test case with a three-phase fault at bus 23 lasting 0.1 s. The total rewards of PARS, MPC, PPO, and UVLS methods in this test case are −823.58, −876.61, −927.34, and −21 901.80, respectively. Fig. 13(A) shows that the voltage with UVLS control (green curve) at bus 33 could not recover to meet the standard (dashed black curve), while with PARS (blue curve), PPO (purple curve), and MPC (green curve) methods, voltages could be recovered to meet the performance standard. Further investigation indicated there were five buses that could not recover their voltage with the UVLS control, while the PARS, PPO, and MPC methods could successfully

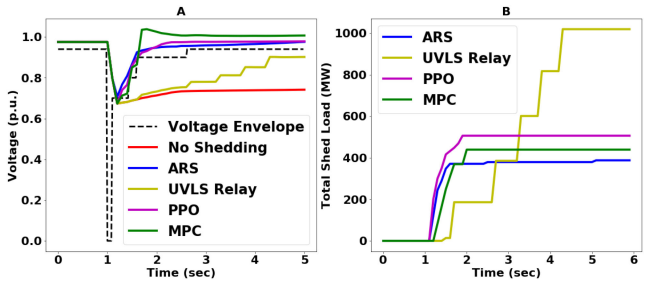

Fig. 13. Test results for a FIDVR event triggered by a three-phase fault at bus 23 of the IEEE 300-bus system. (A) Voltage of bus 33. The dash line denotes the performance requirement for voltage recovery. (B) Total load shedding amount.

recover voltage at all buses above the required levels. Fig. 13(B) shows that PARS shed the least amount of loads among the four methods.

V. CONCLUSION

As described in this paper, a novel derivative-free DRL algorithm named PARS was developed and tailored for power system voltage stability control using load shedding. PARS has the advantages that is more robust against the exploding gradient issue, exhibits better exploration behavior, and is highly scalable and parallelizable.

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 demonstrated that LSTM performs better than FNN in terms of computational efficiency and that PARS scales almost linearly up to more than 1000 CPU cores with no scaling performance saturation. 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). The test results on the IEEE 300-bus system showed: 1) compared with MPC, PARS has a notable advantage in the solution time while achieving comparable results; 2) compared with the state-of-the-art parallel PPO method, PARS has strengths in faster convergence while achieving better solutions, as well as robustness to random seeds in training.

There are several important future research directions:

1) The results in this paper show good potential for PARS to be applied in larger power systems and for different control applications such as frequency control [36] and cascading event mitigation [37].
2) In order to better understand the strength and weakness of different DRL methods and their applicability for various grid control problems, it is important to comprehensively benchmark them through open platforms and standardized test cases.
3) As power systems constantly change, ensuring the adapability of the machine learning model is a requirement

for practical acceptance of machine learning applications [38]. One potential approach is to incorporate meta-learning into DRL to exploit past learning experience to quickly adapt to new operation conditions or learn new tasks [39].

4) Power grid is a mission-critical infrastructure, thus one main concern of application of machine learning (including DRL) for controlling such a critical system is safety and security. Safe RL has been an important research topic and most of previous efforts in this area focus on safe exploration during training [40], [41]. Whereas in power system, the training is mostly done in simulators and the real concern is safety and robustness of the DRL-based control after deployment. Thus, it is important to investigate and solve safety- and robustness-related issues in DRL-based grid control.

REFERENCES

[1] Australian Energy Market Operator, "Black system South Australia 28 Sep. 2016 - Final Report," Mar. 2017. Accessed: June 20, 2020. [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*. New York, NY, USA: Springer Sci. Bus. Media, 2007.

[3] T. Van Cutsem and C. Vournas, "Emergency voltage stability controls: An overview," in *Proc. IEEE Power Eng. Soc. Gen. Meeting*, 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 Trans. Smart Grid*, vol. 11, no. 2, pp. 1171–1182, Mar. 2020.

[5] Y. Dong, X. Xie, K. Wang, B. Zhou, and Q. Jiang, "An emergency-demand-response based under speed load shedding scheme to improve short-term voltage stability," *IEEE Trans. Power Syst.*, vol. 32, no. 5, pp. 3726–3735, Sep. 2017.

[6] A. Stooke and P. Abbeel, "Accelerated methods for deep reinforcement learning," 2018, *arXiv:1803.02811*.

[7] D. Lefebvre, C. Moors, and T. Van Cutsem, "Design of an undervoltage load shedding scheme for the hydro-quebec system," in *Proc. IEEE Power Eng. Soc. Gen. Meeting*, vol. 4, 2003, pp. 2030–2036.

[8] S. Misra, L. Roald, M. Vuffray, and M. Chertkov, "Fast and robust determination of power system emergency control actions," 2017, *arXiv:1707.07105*.

[9] L. Jin, R. Kumar, and N. Elia, "Model predictive control-based real-time power system protection schemes," *IEEE Trans. Power Syst.*, vol. 25, no. 2, pp. 988–998, May 2010.

[10] T. Amraee, A. Ranjbar, and R. Feuillet, "Adaptive under-voltage load shedding scheme using model predictive control," *Electric Power Syst. Res.*, vol. 81, no. 7, pp. 1507–1513, 2011.

[11] M. Glavic *et al.*, "See it fast to keep calm: Real-time voltage control under stressed conditions," *IEEE Power Energy Mag.*, vol. 10, no. 4, pp. 43–55, Jul. 2012.

[12] A. R. R. Matavalam and V. Ajjarapu, "Pmu-based monitoring and mitigation of delayed voltage recovery using admittances," *IEEE Trans. Power Syst.*, vol. 34, no. 6, pp. 4451–4463, Nov. 2019.

[13] H. Sun *et al.*, "Review of challenges and research opportunities for voltage control in smart grids," *IEEE Trans. Power Syst.*, vol. 34, no. 4, pp. 2790–2801, Jul. 2019.

[14] I. 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 Trans. Power Syst.*, vol. 25, no. 3, pp. 1611–1619, Aug. 2010.

[15] 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 Trans. Ind. Informat.*, vol. 17, no. 1, pp. 699–709, Jan. 2021.

[16] 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.

[17] M. Glavic, "(deep) reinforcement learning for electric power systemcontrol and related problems: A short review and perspectives," *Annu Rev. Control*, vol. 48, no. 1, pp. 22-35, 2019.

[18] 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 *Proc. IEEE Innov. Smart Grid Technol.-Asia*, 2018, pp. 13–18.

[19] Z. Yan and Y. Xu, "A multi-agent deep reinforcement learning method for cooperative load frequency control of multi-area power systems," *IEEE Trans. Power Syst.*, vol. 35, no. 6, pp. 4599–4608, Nov. 2020.

[20] C. Chen, M. Cui, F. F. Li, S. Yin, and X. Wang, "Model-free emergency frequency control based on reinforcement learning," *IEEE Trans. Ind. Informat.*, vol. 17, no. 4, pp. 2336–2346, Apr. 2021.

[21] E. Liang *et al.*, "Rllib: Abstractions for distributed reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 3053–3062.

[22] A. Irpan, "Deep reinforcement learning doesn't work yet," Internet 2018, [Online]. Available: https://www. alexirpan. com/2018/02/14/rl-hard.html

[23] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger, "Deep reinforcement learning that matters," 2017, *arXiv:1709.06560*.

[24] C. Taylor, "Concepts of undervoltage load shedding for voltage stability," *IEEE Trans. Power Del.*, vol. 7, no. 2, pp. 480–488, Apr. 1992.

[25] H. Bai and V. Ajjarapu, "A novel online load shedding strategy for mitigating fault-induced delayed voltage recovery," *IEEE Trans. Power Syst.*, vol. 26, no. 1, pp. 294–304, Feb. 2011.

[26] R. S. Sutton and A. G. Barto, *Introduction to Reinforcement Learning*. Cambridge, MA, USA: MIT Press Cambridge, 1998, vol. 135.

[27] V. François-Lavet, P. Henderson, R. Islam, M. G. Bellemare, and J. Pineau, "An introduction to deep reinforcement learning," 2018, *arXiv:1811.12560*.

[28] H. Mania, A. Guy, and B. Recht, "Simple random search of static linear policies is competitive for reinforcement learning," in Proc. *Adv. Neural Inf. Process. Syst.*, 2018, pp. 1800–1809.

[29] T. Salimans, J. Ho, X. Chen, S. Sidor, and I. Sutskever, "Evolution strategies as a scalable alternative to reinforcement learning," 2017, *arXiv:1703.03864*.

[30] R. Huang *et al.*, "Faster than real-time dynamic simulation for large-size power system with detailed dynamic models using high-performance computing platform," in *Proc. IEEE Power Energy Soc. Gen. Meeting*, 2017, pp. 1–5.

[31] Exelon Transmission Planning Criteria, PJM Transm. Plan. Dept.,Valley Forge, PA, USA, 2009.

[32] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997. [Online]. Available: https://doi.org/10.1162/neco.1997.9.8.1735

[33] P. Moritz *et al.*, "Ray: A distributed framework for emerging AI applications," in *Proc. 13th USENIX Symp Oper. Syst. Des. Implement*. (OSDI 18), pp. 561-577. 2018.

[34] RLGC (version 1.0), Accessed: June 10, 2020. [Online]. Available: https://github.com/RLGC-Project/RLGC

[35] M. Plappert *et al.*, "Parameter space noise for exploration," 2017, *arXiv:1706.01905*.

[36] Z. Yan and Y. Xu, "Data-driven load frequency control for stochastic power systems: A deep reinforcement learning method with continuous action search," *IEEE Trans. Power Syst.*, vol. 34, no. 2, pp. 1653–1656, Mar. 2019.

[37] M. R. Almassalkhi and I. A. Hiskens, "Model-predictive cascade mitigation in electric power systems with storage and renewables-part ii: Case-study," *IEEE Trans. Power Syst.*, vol. 30, no. 1, pp. 78–87, Jan. 2015.

[38] L. Duchesne, E. Karangelos, and L. Wehenkel, "Recent developments in machine learning for energy systems reliability management," *Proc. IEEE*, vol. 108, no. 9, pp. 1656–1676, 2020.

[39] W. Yu, J. Tan, Y. Bai, E. Coumans, and S. Ha, "Learning fast adaptation with meta strategy optimization," *IEEE Robot. Automat. Lett.*, vol. 5, no. 2, pp. 2950–2957, Apr. 2020.

[40] G. Dalal, K. Dvijotham, M. Vecerik, T. Hester, C. Paduraru, and Y. Tassa, "Safe exploration in continuous action spaces," 2018, *arXiv:1801.08757*.

[41] J. Garcia and F. Fernández, "A comprehensive survey on safe reinforcement learning," *J. Mach. Learn. Res.*, vol. 16, no. 1, pp. 1437–1480, 2015.