This is the Pre-Published Version.

The following publication R. Wang, X. Bi, S. Bu and Z. Tang, "Deep Reinforcement Learning Approach for Dynamic Distribution Network Reconfiguration Based on Sequential Masking," in IEEE Transactions on Neural Networks and Learning Systems, vol. 36, no. 10, pp. 19270-19284, Oct. 2025 is available at https://doi.org/10.1109/TNNLS.2025.3574208.

Deep Reinforcement Learning Approach for Dynamic Distribution Network Reconfiguration Based on Sequential Masking

Ruoheng Wang, Xiaowen Bi, Siqi Bu, *Senior Member*, *IEEE*, Zhixian Tang

***Abstract*—Dynamic distribution network reconfiguration (DDNR) is a widely used technique for the secure and economic operation of power distribution networks (PDNs), especially in the presence of high-penetration renewable energy sources (RESs). DDNR is realized by controlling the on/off status of remotely controlled switches (RCSs) equipped at power lines in PDNs to optimize power flows. Thanks to the enhanced data availability of PDNs, data-driven solutions to DDNR, such as deep reinforcement learning (DRL), have gained growing attention recently. However, DDNR solves a sequence of combinatorial problems featuring a vast and sparse action space incurred by a so-called "radiality constraint," which is highly challenging for DRLs to handle. Existing DRL methods either are unscalable to large-scale problems or potentially restrict optimality. Hence, we propose a sequential masking strategy to decompose its complex action space into a sequence of maskable sub-action spaces. A GRU-based agent and an adapted soft actor critic (SAC) algorithm are designed accordingly, producing a data-efficient, safety-guaranteed, and scalable DRL solution to the DDNR problem. Comprehensive comparisons with existing data-driven methods and model-based benchmarks are conducted via various case studies, demonstrating the advantages of the proposed method in both algorithmic performance and scalability.**

***Index Terms*—Dynamic distribution network reconfiguration (DDNR), deep reinforcement learning (DRL), sequential masking, soft actor critic (SAC).**

Manuscript received xx xx, xxxx; revised xx xx, xxxx, xx xx, xxxx; accepted xx xx, xxxx. This work was supported in part by the PolyU for RISE Seed Project under Grant U-CDC8, in part by the PolyU for PReCIT Seed Project under Grant 1-CE16, in part by the PolyU for Intra-Faculty Interdisciplinary Project under Grant 1-WZ4L, and in part by the UIC Start-up Research Fund under Grant UICR0700116-25. *(Corresponding author: Siqi Bu)*.

Ruoheng Wang is with Department of Electrical and Electronic Engineering, The Hong Kong Polytechnic University, Kowloon, Hong Kong, China (e-mail: iswangruoheng@foxmail.com).

Xiaowen Bi is with Guangdong Provincial/Zhuhai Key Laboratory of IRADS, and Department of Statistics and Data Science, Beijing Normal-Hong Kong Baptist University, Zhuhai, China (e-mail: xiaowenbi@uic.edu.cn).

Siqi Bu is with Department of Electrical and Electronic Engineering, Shenzhen Research Institute, Research Centre for Grid Modernisation, Research Institute for Smart Energy, Policy Research Centre for Innovation and Technology, International Centre of Urban Energy Nexus, and Centre for Advances in Reliability and Safety, The Hong Kong Polytechnic University, Kowloon, Hong Kong (e-mail: siqi.bu@polyu.edu.hk).

Zhixian Tang is with Department of Electrical and Electronic Engineering, The Hong Kong Polytechnic University, Hong Kong, China (e-mail: zhi-xian.tang@connect.polyu.hk).

Color versions of one or more figures in this article are available at http://ieeexplore.ieee.org.

Digital Object Identifier xxxx

I. Introduction

THE power distribution networks (PDNs) increasingly integrate renewable energy sources (RESs) to promote self-sustainable and low-carbon transitions. High-penetration RESs with uncertain power generation are prone to overburden system operation, leading to various operational issues, such as increasing operation costs and potential risks of over/under-voltage [1]. As an effective and low-cost technique, dynamic distribution network reconfiguration (DDNR) can be incorporated into the daily operation scheme in PDNs for its various functions like power loss reduction [2], [3], system constraint management [4], and stability enhancement [5]. PDNs are required to operate in a radial topology to facilitate management [6], and therefore DDNR allows PDNs to operate in a different configuration (leading to a different topology) by controlling the on/off status of remotely controlled switches (RCSs) equipped at power lines. Hence, DDNR aims to determine the optimal configuration in real time for the varying operation conditions of PDNs to enhance interested system indicators for operation [7].

Existing studies for DDNR are primarily based on mathematical programming [7], [8] and heuristic algorithms [9], [10]. While these methods may be capable of solving DDNR problems, they entail complete observability of system parameters in real time, leading to expensive implementation in practice. As the model-free alternative, the data-driven solution has been developed recently for DDNR applications by leveraging promising deep reinforcement learning (DRL). Owing to the combinatorial nature and a special "radiality constraint" of DDNR problems, the vast and constrained action space of DDNR poses challenges for DRLs in learning competitive policy with model-based traditional methods. The "radiality constraint" requires a DRL agent to act safely to avoid infeasible configurations of RCSs that cause disconnection or looped connections in PDNs [11]. Existing solutions have addressed this constraint by 1) penalizing infeasible actions via rewards [12], 2) constructing action sets for all feasible configurations [13], and 3) adopting a reduced action space [3]. However, these strategies are known to either lack scalability to large-scale problems or limit optimality.

To tackle the identified research gap, this paper proposes an action space-unrestricted and scalable learning framework for DDNR problems. Concretely, a sequential masking strategy is proposed to decompose the DDNR action into a sequence of sub-actions. Starting from a fully connected graph of the PDN,

© 2025 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

each sub-action decides which RCS to stay open sequentially, after which an arbitrary configuration can be determined. A computationally scalable mask is then introduced to truncate the probability of those constraint-violated sub-actions. The sequential decision process is formulated as a probability chain and modeled by the gated recurrent unit (GRU) thanks to its recurrent neural structure. Lastly, the necessity of modeling the truncated policy distribution and the desire for data efficiency inspire us to adapt the policy-gradient and off-policy SAC to the designed sequential masking process. SAC is applied to the discrete action space via Gumbel softmax and is trained on a dedicated representation of configurations, enabling a scalable and data-efficient algorithm for DDNR problems. Various case studies are implemented to compare the proposed method with the existing action strategies and model-based approaches on various PDNs. Results show that our method can achieve competitive performance and is scalable to the large-scale problem as a model-free solution.

The major contributions are summarized below.

1) It is the first paper to summarize the pros and cons of existing action strategies for tackling the vast and constrained action space of DDNR. Given the identified gap, a sequential masking strategy is proposed for decomposing its action space into a sequence of maskable sub-action spaces, which are scalable, unrestricted, and safe for DRL to learn efficiently.

2) An off-policy DRL algorithm is proposed to demonstrate the viability of the proposed strategy, where a GRU agent is designed to structurally model the probability chain of the sequential decision process based on a dedicated binary-vector representation of configurations, and SAC is adapted to its discrete action space for sample-efficient training.

3) Extensive case studies are conducted to validate our method with various action strategies and model-based benchmarks, revealing that our method can surpass existing DRL solutions in either scalability or performance for DDNR.

The rest of the paper is organized as follows: Section II introduces the related work. Section III introduces the problem of DDNR and its formulation as an MDP. Section IV summarizes the previous DRL approaches and presents the proposed sequential masking strategy and algorithm for DDNR. Section V validates the proposed method in various case studies. Section VI draws our conclusions.

II. Related Work

The section summarizes the advantages and shortcomings of existing methods for DDNR problems. Existing methods can be divided into model-based and model-free methods, which are introduced as follows.

Model-based methods: They constitute the majority of traditional methods for DDNR problems, including some naïve methods [1], heuristic methods [9], and mathematical programming [7]. Naïve methods usually resort to repetitively solving power flows to assess different network configurations generated by a simple strategy. Such strategies fall short of efficiently utilizing computation resources to find promising configurations. To improve the search process, meta heuristics, such as genetic algorithms [9] and harmony search [10], have been developed to seek quality solutions for multiple objectives in DDNR problems [4]. The black-box optimization of these meta heuristics allows them to be easily applied to DDNR problems regardless of problem complexity and non-linearity. However, they are still inefficient for online decision-making as they rely on power flow calculations for objective evaluation [11]. With a proper approximation of the physical model of PDNs, mixed-integer programming (MIP) [7], [14], [15] can be used to solve DDNR problems with efficiency and robustness, though it may be computationally costly for very large-scale problems and suffer approximation errors pertaining to problem complexity. Overall, these model-based methods are more appealing if the complete parameters of PDNs are available for the real-time decision-making of DDNR, while such a pre-condition may result in higher operation costs and reduce their reliability under contingency.

Model-free methods: They are growing popular due to recent advances in data-driven techniques, particularly, deep learning and reinforcement learning. For instance, in [5] a deep learning model is introduced into DDNR to realize rapid control response to abrupt voltage contingency via the generalizability of neural networks (NNs). Without relying on expert knowledge like supervised training, DRL algorithms, such as deep Q network (DQN) [4], [12] and soft actor critic (SAC) [3], can learn high-quality policy from scratch by interacting with a simulated DDNR environment. As reported in DDNR literature, DRL methods have several advantages over model-based counterparts, such as rapid decision-making via generalization [16], strong predictability and modeling capability for uncertain and complex operation scenarios of PDNs [2], [13], and model-free nature for reducing the cost of overseeing entire PDNs [3]. However, the radiality constraint in DDNR creates a vast and sparse discrete action space, which is still not soundly addressed by existing DRL methods. In contrast to the model-based MIP that can effectively address the constraint via "spanning tree" modeling [7], it is difficult for DRL to integrate the hard constraint into its learning process directly. Existing research addresses the challenge as follows. 1) Studies [12], [17], [16], [18] penalize violated actions by adding punitive signals in the reward function. However, the scarcity of feasible actions can lead to inefficient learning, and the agent may still generate dangerous actions (a backup plan in [16] may help guarantee its safety). 2) Studies [13], [19], [20], [21] construct a valid action set before training for the agent to pick feasible actions, but its scalability is questionable, especially considering the sheer number of feasible configurations in large PDNs. 3) Studies [3], [11], [22], [23] adopt a scalable solution that restricts action space by allowing only a pair of RCSs to exchange status (i.e., close one RCS and then open one), and thus, the action mask can be analytically constructed and imposed on the agent to eliminate infeasible options to the radiality constraint. However, such an action space reduction inevitably limits the optimality of DDNR. To tackle the identified research gap, this paper develops an action space-unrestricted, scalable, and safe DRL framework for DDNR problems in the following sections.

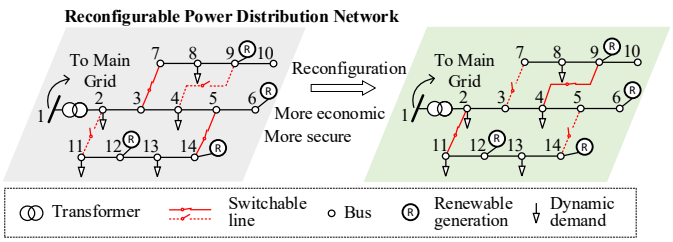


Fig. 1. Power distribution network with remotely controlled switches.

III. Preliminary

The section introduces the typical modeling of a DDNR problem involving its objectives and constraints, which are then formulated into an MDP to facilitate DRL methods.

A. Reconfigurable Power Distribution Network

PDNs normally operate in a radial topology to facilitate management, as illustrated in Fig. 1. Electrical power is delivered from the high-voltage main grid to the low-voltage PDN and finally to the demand sites (nodes in Fig. 1) via transformers and underground cables (lines in Fig. 1). Due to high-penetration RESs distributed in the PDN, system dynamics are growingly complicated in both spatial and temporal dimensions. For enhanced network security and economy, shown in Fig. 1, DDNR can be adopted to mitigate various operational issues in PDNs, such as overloading and overvoltage. DDNR is enabled by the two types of RCSs in PDN: the sectionalizing switch (normally closed) and the tie switch (normally open). DDNR aims to form optimal radial configurations for PDNs by determining the on/off status of RCSs in real time.

The PDN can be described as a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is a set of electric nodes and $\mathcal{E}$ is a set of existing lines. Power flows follow the physical rule below.

$$\begin{aligned} P_{i,t} &= V_{i,t}\sum_{j\in\mathcal{V}} b_{ij,t}V_{j,t}\left(G_{ij}\cos\theta_{ij,t} + B_{ij}\sin\theta_{ij,t}\right) \\ Q_{i,t} &= V_{i,t}\sum_{j\in\mathcal{V}} b_{ij,t}V_{j,t}\left(G_{ij}\sin\theta_{ij,t} - B_{ij}\cos\theta_{ij,t}\right) \end{aligned}, \quad (1)$$

where $P_{i,t}$ and $Q_{i,t}$ are the active and reactive power injected into the node i at time t, respectively, $V_{i,t}$ is the voltage magnitude, $\theta_{ij,t}$ is the phase angle between $V_{i,t}$ and $V_{j,t}$, $b_{ij,t}$ is a binary variable indicating the service status of the line ij, G_{ij} and B_{ij} are the line conductance and susceptance, respectively. Nodal power injections consist of generation and demand that follow $P_{i,t} = P_{i,t}^g - P_{i,t}^d$ and $Q_{i,t} = Q_{i,t}^g - Q_{i,t}^d$, where g and d stand for generation and demand, respectively. The generation of RESs is bounded by a rated capacity S_i^g, i.e., $\left(P_{i,t}^g\right)^2 + \left(Q_{i,t}^g\right)^2 \le \left(S_i^g\right)^2$. The power imbalance is compensated by the main grid via the root node that features $V_i = 1$ per unit (p.u.) with $i = 1$. At the system level, the energy balance leads to $\sum_{i\in\mathcal{V}} P_{i,t}^g = \sum_{i\in\mathcal{V}} P_{i,t}^d + P_t^l$, where P_t^l is the line loss. The total power loss is calculated by:

$$P_t^L = \sum_{ij\in\mathcal{E}} P_{ij,t}^l = \sum_{ij\in\mathcal{E}} b_{ij}R_{ij}\frac{P_{ij,t}^2 + Q_{ij,t}^2}{V_{i,t}^2}, \quad (2)$$

where R_{ij} is the line resistance, $P_{ij,t}$ and $Q_{ij,t}$ are the power flowing through the line ij.

During operation, the nodal voltages in PDNs should be maintained to approach a nominal value, i.e., 1 p.u., to guarantee satisfactory power quality and avoid adverse effects on power equipment [6]. To this end, voltages should be regulated by DDNR into a security range [11]:

$$1 - V_B \le V_{i,t} \le 1 + V_B, \quad (3)$$

where $1 - V_B$ and $1 + V_B$ confine the voltage magnitude to a security region, e.g., 0.95-1.05 p.u. [3].

Let $\{b_{ij,t} | ij \in \mathcal{E}_s\}$ with $\mathcal{E}_s \subseteq \mathcal{E}$ be a set of binary variables denoting the on/off status of RCSs. After DDNR, the radiality constraint has to be satisfied [7] by: 1) there is no isolated node in the PDN (leading to power outage); 2) PDN should operate in a radial topology. It is described by:

$$|\mathcal{C}| = 1, \quad (4a)$$

$$|\mathcal{V}| - |\mathcal{C}| = |\mathcal{E}| - \sum_{ij\in\mathcal{E}_s}\left(1 - b_{ij,t}\right), \quad (4b)$$

where $\mathcal{C}$ is a set of connected components in $\mathcal{G}$. In graph theory, a forest graph $\hat{\mathcal{G}}$ always follows $|\hat{\mathcal{V}}| = |\hat{\mathcal{C}}| + |\hat{\mathcal{E}}|$, and it becomes a tree graph if $|\hat{\mathcal{C}}| = 1$. As the RCS is electro-mechanical equipment, a limit on the number of times (NoT) of switching should be considered by DDNR to prolong their service life. The NoT of switching denoted by A_t can be calculated as follows:

$$A_t = \sum_{ij\in\mathcal{E}_s, t\in\mathcal{T}} \left|b_{ij,t} - b_{ij,t-1}\right|, \quad (5)$$

B. Problem Formulation as Markov Decision Process

DDNR can be formulated into a MDP to facilitate DRL learning. MDP can be described as a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P}, \gamma \rangle$, where $s_t \in \mathcal{S}$ is the observed state of the environment, $a_t \in \mathcal{A}$ is the action generated by an agent, $r_t = \mathcal{R}(s_t, a_t)$ is the reward returned by a function $\mathcal{R}$, $s_{t+1} = \mathcal{P}(s_t, a_t)$ is the successive state returned by the transition function $\mathcal{P}$, and γ is the discount factor for better convergence. The critical parts of MDP for the DDNR problem are described as follows.

State: For alleviating the pressure on data acquisition, only nodal power/voltage and switch status are assumed for decision-making, which is defined as follows:

$$s_t = \{V_{i,t}, P_{i,t}, Q_{i,t} | i \in \mathcal{V}, t \in \mathcal{T}_{hist}\} \cup \{b_{ij,t-1} | ij \in \mathcal{E}_s\}, \quad (6)$$

where $\mathcal{T}_{hist} = \{t - t_h, \dots, t\}$ represents a set of historical time steps for the agent to learn temporal patterns.

Action: The action of the DDNR agent is to determine the status of RCSs at time t, i.e., $a_t = \{b_{ij,t} | ij \in \mathcal{E}_s\}$.

Agent: Given s_t, the agent π learns to find a policy for real-time decision, i.e., $a_t = \pi(s_t)$. In DRLs, a parameterized policy π_θ aims to find the optimal θ that maximizes $J(\theta)$, where $J(\theta) = \sum_{t\in\mathcal{T}} \mathbb{E}_{s_t, a_t\sim\tau_\pi} r_t$, where τ_π is the trajectory produced by the agent π.

Transition: During operation, the observed states will evolve with dynamic power demand and RES generation interacting with each other via Eq. (1). The modeling can be realized by open-source platforms like Pandapower [24] with the real-world time series of demand and generation. The complexity of network interaction and the uncertainty in time series make DDNR challenging.

Reward: Algorithmic performance can be sensitive to reward shaping. The voltage barrier function with a bowl shape used in [25] is employed for voltage reward. Given economic considerations of the power loss and switching cost, the reward r_t is defined as follows:

$$\begin{aligned} r_t &= -k_v r_{v,t} - k_e r_{e,t} - r_{c,t} \\ r_{v,t} &= \begin{cases} \left(\frac{V_t^{dev}}{V_B}\right)^2, & \forall\, V_{dev,t} \le V_B \\ 1 + C_v \frac{V_t^{dev} - V_B}{V_B}, & \forall\, V_{dev,t} > V_B \end{cases}, \\ r_{e,t} &= P_t^L C_p + A_t C_a \end{aligned} \tag{7}$$

where $r_{v,t}$ punishes the voltage deviation based on the maximum nodal voltage deviation at the time t, i.e., $V_t^{dev} = \max\{|V_{i,t} - 1| | i \in \mathcal{V}\}$ in p.u., C_v imposes a large penalty on the violated voltage deviation, $r_{e,t}$ is the economic penalty consisting of power loss and action cost in \$, C_p and C_a are electricity and switching prices [3], respectively, and $r_{c,t}$ is a large constant penalty for actions violating the constraint in Eq. (4). RCSs is dispatched in a fixed and slow time interval (e.g., $\Delta t = 1$ hour). Therefore, a daily operation scheme with fixed intervals (e.g., $t \in \mathcal{T} = \{0,1,\dots,24\}$ in hours) is considered to facilitate episodic learning of DRL.

To clarify the DDNR problem more rigorously, the corresponding mathematical formulation of the problem in the form of MIP is provided in the Appendix.

IV. Methodology

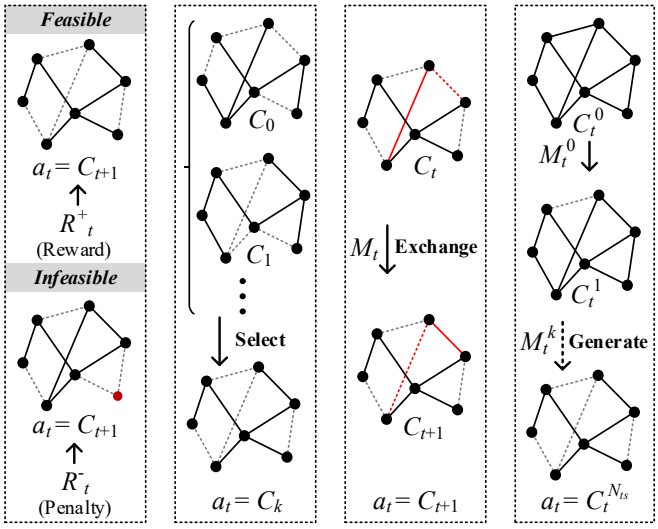
The section elaborates on existing DRL action strategies for DDNR problems and then introduces our proposed method with its theoretical explanation and practical implementation.

A. Sequential Action Masking Strategy

One challenge for DRLs in the DDNR problem is the vast and sparse action space incurred by its topology constraint in Eq. (4). For instance, the action size $|\mathcal{A}|$ of DDNR amounts to a combination number of $2^{|\mathcal{E}_s|}$, while its rate of feasible actions is lower than 5% on a modified IEEE 33-bus PDN [13]. This rate tends to be even lower with growing RCSs [12]. Several commonly used strategies for addressing the action space are illustrated in Fig. 2. Their implementations are summarized below, and our proposed strategy is presented subsequently.

Penalize infeasible actions via rewards: Fig. 2(a) shows an intuitive strategy to tackle the constraint in Eq. (4) by adding an order of magnitude larger penalty to infeasible actions generated by the agent [12]. It enables the agent to sample freely in the original vast action space, whereas it may suffer two major drawbacks: 1) frequent penalty signals due to the scarcity of feasible actions quickly make the policy stuck in a local optimum; 2) "indirect constraint" on the agent likely leads to dangerous behaviors once the agent cannot generalize to unseen states properly in practice.

Encode all the feasible actions: Given the scarcity of feasible actions, Fig. 2(b) suggests directly encoding all the feasible actions into a valid action set that can be constructed via an enumeration process [13]. Based on the action set, algorithms like DQN can be applied to DDNR without adaptation. While this strategy can guarantee a hard constraint for Eq. (4), its scalability to the large-scale DDNR problem is questionable. For instance, even with a rate of 0.1%, the size of the valid action set constructed by a PDN with thirty RCSs can be more than one million, thus making the strategy incompetent to handle large-scale problems.



(a) Penalize infeasible configurations (b) Encode all the feasible configurations (c) Limit changes to configurations (d) **The proposed**: generate feasible configurations

Fig. 2. Four action strategies to produce configurations for DDNR.

Restrict action space: As shown in Fig. 2(c), a scalable solution is to reduce the action space by restricting the extent of topology changes. Concretely, in each decision step, one RCS is selected to close and another RCS to open, so as to exchange their on/off status. The action size is then reduced from an exponential scale of $2^{|\mathcal{E}_s|}$ to a polynomial scale. For instance, assuming that $a_t = (ij_1, ij_2)$ selects a pair of RCSs to close and open, its action size can be $|\mathcal{E}_s|^2 - |\mathcal{E}_s| + 1$ [11], where $|\mathcal{E}_s|^2 - |\mathcal{E}_s|$ is the combination number of (ij_1, ij_2) with $ij_1, ij_2 \in \mathcal{E}_s$ and $ij_1 \neq ij_2$, and the one extra action is the option to maintain the current configuration. The action can strictly follow Eq. (4) using an action mask $M_t \in \mathbb{R}^{|\mathcal{E}_s|^2 - |\mathcal{E}_s| + 1}$. The construction process of M_t can be referred to [11]. In M_t, $M_{t,(ij_1,ij_2)} = 1$ if (ij_1, ij_2) is a feasible action, and $M_{t,(ij_1,ij_2)} = 0$ if otherwise. Hence, a feasible action can be sampled from a truncated conditional probability distribution by:

$$a_t \sim p(C_{t+1} | C_t, M_t, s_t). \tag{8}$$

For instance, it can be modeled in DQN by $a_t = \text{argmax}_{a_t}(Q_\theta(a_t|s_t) + \Gamma(M_t))$, where Q_θ outputs Q values and $\Gamma(M_t)$ is defined as $-C_m(1 - M_t)$ with a large constant C_m (e.g., $C_m = 1e16$). While the approach is more scalable in computation than strategy (2) and more secure in practice than strategy (1), it inevitably restricts the optimality of the policy due to the action space reduction. In DDNR, the number of RCSs to keep open is always a constant of the number of tie switches, i.e., N_{ts}. It has to take up to N_{ts}-step actions to reach an arbitrary configuration. Since DDNR is implemented in a long-time interval, such a restriction on

topology changes may hinder the agent from catching up to the fast-varying dynamics of PDN, especially in large PDNs owning more tie switches.

Proposed sequential masking strategy: A novel action strategy based on a sequential masking process is proposed to fully handle the fundamental constraints in Eq. (4) for DDNR. The strategy is presented in Fig. 2(d), where a feasible configuration is produced by sequentially disconnecting lines (via RCSs) from a fully connected graph of a PDN with the aid of masks. Its rationale is explained as follows.

In the fundamental constraints, constraint (4a) states that the network should be connected (i.e., only one connected component), and constraint (4b) demands that the number of nodes (i.e., $|\mathcal{V}|$) should always be one ($|\mathcal{C}| = 1$) more than the number of connected lines (i.e., $|\mathcal{E}| - \sum_{ij\in\mathcal{E}_s}(1 - b_{ij,t})$). Since constraint (4b) requires that there should be $|\mathcal{E}| - |\mathcal{V}| + |\mathcal{C}|$ disconnected lines after reconfiguration, it can be realized by sequentially switching off this number of RCSs starting from a fully connected graph. However, this cannot necessarily satisfy constraint (4a). Directly penalizing the agent for the constraint can be of low efficiency. Luckily, by decomposing the action space into a sequence of sub-action spaces for constraint 4(b), we can now guarantee constraint 4(a) by ensuring that the agent only switches off one of the RCSs in cycles of the current graph each time. It is obvious that cutting off a line in a cycle will not cause disconnection of a graph, as there are at least two paths to connect any two nodes having the line between them. To force the agent to select RCSs in cycles, action masks can be introduced and constructed based on existing cycles to eliminate the probabilities of infeasible action options. Finally, the sequential masking strategy is finished with "sequential" decision-making for constraint 4(b) and "masking" for constraint 4(a).

Given the sequential masking strategy, for each intermediate configuration C_t^k, a sub-action can be sampled from a mask-truncated sub-policy distribution as follows:

$$a_t^k \sim p(C_t^k | C_t^{k-1}, M_t^{k-1}, s_t), \tag{9}$$

where $k \in \{1, \dots, N_{ts}\}$, $a_t^k \in \mathbb{R}^{|\mathcal{E}_s|}$ is the k^{th} sub-action at time t, and $M_t^k \in \mathbb{R}^{|\mathcal{E}_s|}$ is the k^{th} binary sub-mask.

We now provide a simple yet likely not the most algorithmically efficient implementation for its mask construction: first search for fundamental cycles of the intermediate topology C_t^k, then fill one in M_t^k for those RCSs in the cycles and zero for the rest of RCSs. The computation time of fundamental cycles (also cycle basis) in NetworkX grows with $O(|\mathcal{V}|^\beta)$ bounded by $2 \le \beta \le 3$ [26]. As N_{ts} sub-actions should be conducted, the calculation time of masking grows with $O(|\mathcal{V}|^\beta \cdot N_{ts})$. This can be further improved by merging the neighboring nodes whose connected line is not used for DDNR. Hence, the vertex number is reduced, and the operation time can roughly grow with $O(|\mathcal{E}_s|^\beta \cdot N_{ts})$.

Note that with recurrent calculation for generating sub-policy and increased complexity for calculating masks compared with the strategy in Fig. 2(c), our strategy

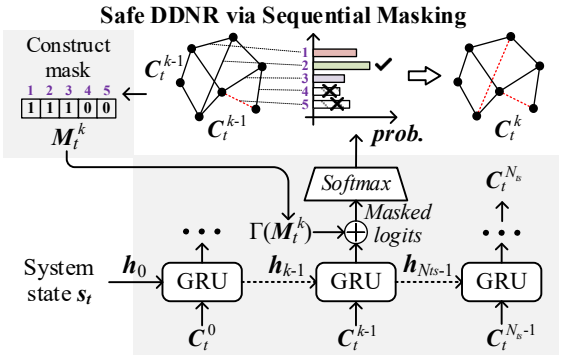


Fig. 3. Sequential masking process in a decision step.

inevitably takes longer for training and inference. Nevertheless, following offline training and online inference, DRL can isolate such a computational burden from online decision-making. Commonly, there is enough time budget for offline training, and the increased inference time (in milliseconds) can easily match the implementation time interval of DDNR (in tens of minutes to a few hours [11]). Overall, our proposed strategy can guarantee both the reachability and safety of actions for DDNR and is computationally viable for large-scale PDNs.

B. GRU-based Sequential Masking Process

The proposed method for flexibly generating network configurations employs a sequence of sub-actions made in an action step of DDNR, which can be described by a probability chain rule below.

$$p(C_{t+1} = C_t^{N_{ts}} | s_t) = \prod_{k=1}^{N_{ts}} p(C_t^k | C_t^{k-1}, M_t^{k-1}, s_t), \tag{10}$$

where C_t is a set of the current status of RCSs, C_t^k is a set of intermediate status of RCSs for calculation, a_t directly applies the last configuration $C_t^{N_{ts}}$ after a sequence of sub-actions delinking the edges from an initial configuration C_t^0, where all the lines are assumed to be connected. Note that C_t^k is just a feature in calculation and does not require actual implementation in the PDN, which will be detailed in Section III-C. Then, we demonstrate how an RNN-based agent can be constructed accordingly to generate a sequence of sub-policy distributions conditioned on the current state s_t and the previous intermediate configuration C_t^{k-1}.

Recurrent neural networks (RNNs) are known for their efficiency in sequential modeling. For instance, an RNN-based seq-to-seq framework is used in Pointer Network [27] to solve combinatorial optimization problems. In this paper, GRU, as a widely used variant of RNNs, is employed to generate a sequence of conditional probabilities defined in Eq. (10). The calculation in a GRU cell can be briefly described by:

$$\begin{aligned} r_k &= \sigma(W_r \cdot [h_{k-1}, x_k]^T) \\ z_k &= \sigma(W_z \cdot [h_{k-1}, x_k]^T) \\ \hat{h}_k &= \tanh\left(W_h \cdot [r_k \otimes \hat{h}_{k-1}, x_k]^T\right), \\ h_k &= (1 - z_k) \otimes h_{k-1} + \hat{h}_k \otimes z_k \end{aligned} \tag{11}$$

where $\otimes$ denotes element-wise product, r_k and z_k serve as the reset and update gates at the k^{th} step, respectively, to control

information flows delivered by input x_k and memory h_{k-1}, and W_r, W_z, and W_h are optimizable parameters. A GRU cell outputs h_k for the next GRU cell and downstream neural network (NN) modules. The calculation is simplified as $h_k = \mathrm{GRU}(h_{k-1}, x_t)$. By conditioning h_0 on s_t, h_k can be used to pass state information for subsequent decisions. The configuration C_t^{k-1} can be used as the input x_k. Accordingly, the function $p_\theta(C_t^k | C_t^{k-1}, s_t; \theta)$ can be parameterized by:

$$C_t^k = g(\mathrm{GRU}(h_{k-1}, C_t^{k-1}), M_t^{k-1}), \quad (12)$$

where $g(\cdot)$ denotes a series of operations to convert the GRU output h_k into the next C_t^k using the mask M_t^{k-1}.

The sequential masking process is illustrated in Fig. 3. The truncation of the probability distribution for infeasible immediate configurations can be described as follows:

$$p_\theta(C_t^k | C_t^{k-1}, M_t^{k-1}, s_t) = \mathrm{Softmax}\left(l_t^k + \Gamma(M_t^{k-1})\right), \quad (13)$$

where $l_t^k \in \mathbb{R}^{|\mathcal{E}_s|}$ denote the logits output by the last layer of NNs at the current step k (assume NNs to be stacked on GRUs), and $\Gamma(M_t^{k-1})$ sets the invalid logits (corresponding to an infeasible sub-action) in l_t^k into extremely negative values while valid logits remain unchanged. Given the truncated policy distribution via Softmax, a sampling policy can then be applied to pick a safe configuration, guaranteeing the flexibility of action to reach any feasible C_{t+1} in the one-step decision regardless of the current configuration C_t.

C. Off-policy DRL for Configuration Generation.

DRL algorithms used for solving DDNR problems are mostly model-free and off-policy algorithms. These methods can be majorly categorized into value-based (e.g., DQN [12]) and actor-critic-based DRLs (e.g., SAC [3]). We develop an SAC-based neural agent for the proposed sequential masking strategy considering that: 1) a policy-based agent that outputs policy distribution is naturally more suitable for the proposed sequential masking strategy than a value-based agent; 2) SAC with entropy regularization for balancing exploration and exploitation can be advantageous in algorithmic performance and convergence stability for DDNR problems [11]; 3) off-policy SAC has a better sample efficiency than on-policy algorithms that can also be adapted to the strategy, such as proximal policy optimization.

Following soft policy iteration, SAC aims to simultaneously maximize the entropy of the policy π and the reward $r_t = \mathcal{R}(s_t, a_t)$ as follows:

$$\max_\pi J(\pi) = \sum_{t \in \mathcal{T}} \mathbb{E}_{s_t, a_t \sim \tau_\pi}\left[\mathcal{R}(s_t, a_t) + \alpha \mathcal{H}(\pi(\cdot|s_t))\right], \quad (14)$$

where $\mathcal{H}(\pi(\cdot|s_t)) = -\sum_{a_t} \pi(a_t|s_t) \log \pi(a_t|s_t)$ calculates the entropy of π given s_t, τ_π denotes the trajectory generated by π, and α adjusts the weight of the entropy. The soft policy iteration of SAC can be described by:

$$\begin{aligned} Q_\pi(s_t, a_t) &\leftarrow r_t + \mathbb{E}_{s_{t+1} \sim \tau_\pi} V_\pi(s_{t+1}) \\ V_\pi(s_t) &\leftarrow \mathbb{E}_{a_t \sim \tau_\pi}\left[Q_\pi(s_t, a_t) + \alpha \mathcal{H}(\pi(\cdot|s_t))\right], \end{aligned} \quad (15)$$

where V_π and Q_π are the state and state-action value functions, respectively. Let the critic Q_π be parameterized by ϕ and denoted as Q_ϕ, and let the actor π be parameterized by θ and denoted as π_θ. SAC trains the critic using a time-difference (TD) loss and the actor using policy gradients as follows [28]:

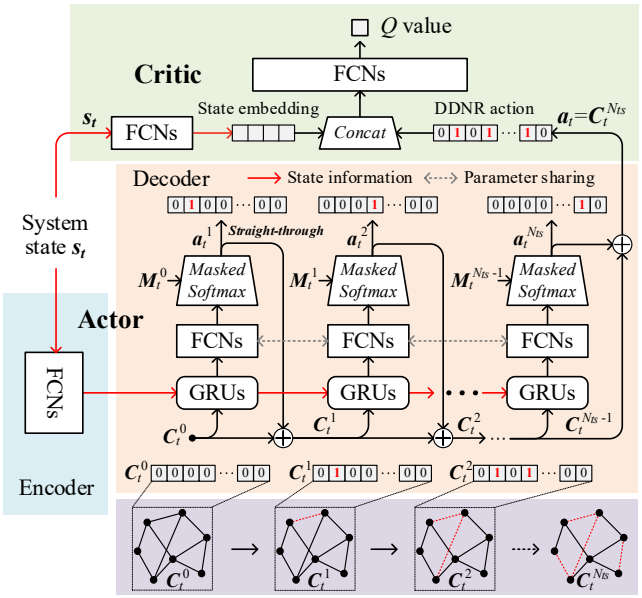


Fig. 4. The actor-critic neural structure of the SAC agent for generating arbitrary feasible configurations via a sequential masking process.

$$\begin{aligned} &\nabla_{\phi_i} J_Q(\phi_i) = \nabla_{\phi_i} \left| y_t - Q_{\phi_i}(s_t, a_t) \right|^2, i \in \{1,2\} \\ &y_t = r_t + \gamma \min_{i=1,2} \left(Q_{\hat{\phi}_i}(s_{t+1}, a_{t+1}) + \alpha \mathcal{H}(\pi_\theta(\cdot|s_{t+1})) \right), \end{aligned} \quad (16)$$

$$\nabla_\theta J_\pi(\theta) = -\nabla_\theta \alpha \mathcal{H}(\pi_\theta(\cdot|s_t)) - \nabla_{a_t} Q_{\phi_1}(s_t, a_t) \nabla_\theta \pi_\theta(a_t|s_t), \quad (17)$$

where y_t is the TD target. J_Q adopts a double DQN (DDQN)-style TD loss by maintaining two delayed updated critics $\left\{ Q_{\hat{\phi}_i} \middle| i \in \{1,2\} \right\}$ to suppress value over-estimation, J_π passes policy gradients from the critic Q_{ϕ_1} into the actor π_θ.

The proposed strategy based on SAC is illustrated in Fig. 4, where the state s_t is first extracted by the fully connected networks (FCNs) and then fed into the actor and critic, respectively. During calculation, the current configuration C_t is defined as a binary vector $\left[b_{ij,t} \middle| ij \in \mathcal{E}_s \right]^T$, the intermediate configuration C_t^k is a binary vector $\left[1 - b_{ij,t}^k \middle| ij \in \mathcal{E}_s \right]^T$ to represent the status of RCSs, and the initial configuration C_t^0 is a vector $\left[1 - b_{ij,t}^0 \middle| b_{ij}^0 = 1, ij \in \mathcal{E}_s \right]^T$ with all zeros to represent a completely connected graph of a PDN at the outset. Such an arrangement allows us to represent the sub-action a_t^k as a one-hot vector, which is important for subsequent operations. The one in a_t^k indicates which RCS is the next to open based on C_t^{k-1}. Hence, once a sub-action a_t^k is obtained, the next intermediate configuration C_t^k is calculated by:

$$C_t^k = C_t^{k-1} + a_t^k. \quad (18)$$

The simple additive operation allows gradients to be propagated through forward calculation flows in agents. The final DDNR action, i.e., the configuration $C_t^{N_{ts}}$, can be obtained by summing up all the sub-actions:

$$a_t = C_t^{N_{ts}} = \sum_{k=1}^{N_{ts}} a_t^k. \quad (19)$$

Algorithm Sequential-masking SAC for DDNR problems

Parameter: N_e – number of episodes, N_s – number of gradient steps, N_{ts} – number of tie switches, ρ – Polyak averaging factor, η_Q, η_π, η_α – learning rates, μ – annealing rate.

Input: θ and $\left\{\phi_i, \hat{\phi}_i \middle| i \in \{1,2\}\right\}$ – parameters of the agent.

Output: π_θ for online execution.

Initialize replay buffer $\mathcal{D} \leftarrow \emptyset$ and target critics $\hat{\phi}_i \leftarrow \phi_i$.
for $n \leftarrow 1$ to N_e **do**
  Reset DDNR environment and obtain s_1.
  for $t \leftarrow 1$ to $|\mathcal{T}|$ **do**
    Construct C_t^0 and calculate h_t^0 from s_t.
    for $k \leftarrow 1$ to N_{ts} **do**
      Construct sub-mask M_t^k.
      Calculate $(l_t^k, h_t^k) \leftarrow \pi_\theta(h_t^{k-1}, C_t^{k-1})$.
      Sample sub-action $a_t^k \leftarrow \text{GumbelSoftmax}(l_t^k + \log M_t^k)$.
      Construct configuration $C_t^k \leftarrow C_t^{k-1} + a_t^k$.
    end
    Obtain action $a_t \leftarrow \sum_k^{N_{ts}} a_t^k$.
    Execute $s_{t+1} \leftarrow \mathcal{P}(s_t, a_t)$ and $r_t \leftarrow \mathcal{R}(s_t, a_t)$.
    Save transition by $\mathcal{D} \leftarrow (s_t, a_t, r_t, s_{t+1}) \cup \mathcal{D}$.
  end
  for $i \leftarrow 1$ to N_s **do**
    Sample mini-batch $\mathcal{B}\sim\text{Uniform}(\mathcal{D})$.
    Update critics by $\phi_i \leftarrow \phi_i - \frac{\eta_Q}{|\mathcal{B}|}\nabla_{\phi_i} J_Q(\phi_i)$.
    Update target critics by $\hat{\phi}_i \leftarrow \rho\phi_i + (1-\rho)\hat{\phi}_i$.
    Update actor by $\theta \leftarrow \theta - \frac{\eta_\pi}{|\mathcal{B}|}\nabla_\theta J_\pi(\theta)$.
    Update α by $\alpha \leftarrow \alpha - \frac{\eta_\alpha}{|\mathcal{B}|}\nabla_\alpha J_\alpha(\alpha)$.
  end
  Anneal target $\bar{H}$ by $\bar{H} \leftarrow \mu \cdot \bar{H}$.
end

One can simply use $C_{t+1} = 1 - a_t$ to implement a_t. With this differentiable masking process, the GRU-based agent can explore the complex action space safely and scalably and can be optimized efficiently by back-propagation algorithms.

There are two adaptations of SAC to the designed learning framework. First, SAC was originally proposed for continuous actions [28], whereas the agent in Fig. 4 features a discrete and one-hot action representation. Hence, a re-parameterization trick called Gumbel softmax [29] is adopted to build a discrete and stochastic policy for the actor and approximate policy gradients delivered from the critic. The mask-truncated Gumbel softmax distribution is formulated as follows:

$$p_\theta(C_t^k|C_t^{k-1}) = \text{softmax}\left((l_t^k + \Gamma(M_t^k) + \epsilon)\cdot\tau_g^{-1}\right), \quad (20)$$
$$\epsilon = -\log(-\log u),\ u\sim\text{Uniform}(0,1)$$

where l_t^k is output from the last layer of NNs stacked on the GRUs, as shown in Fig.4., ϵ is the sampled value from the Gumbel distribution, and τ_g is the temperature coefficient. Gumbel softmax operation proves to be equal to sampling from the corresponding softmax distribution, but the operation is differentiable. A straight-through conversion [29] can be used to generate actions in one-hot representation by:

$$a_t^k = \text{const}\left(z - p_\theta(C_t^k|C_t^{k-1})\right) + p_\theta(C_t^k|C_t^{k-1}), \quad (21)$$

where $\text{const}(\cdot)$ cuts off gradient passing, and z represents the one-hot vector obtained by $z = \text{argmax}_{a_t^k}\left(p_\theta(C_t^k|C_t^{k+1})\right)$. With the conversion, a differentiable binary-vector operation in Eqs. (18) and (19) can be realized.

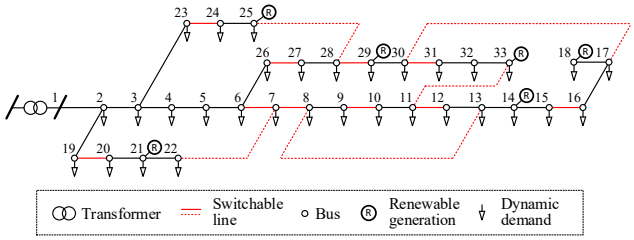


Fig. 5. A modified IEEE 33-bus PDN for the DDNR problem.

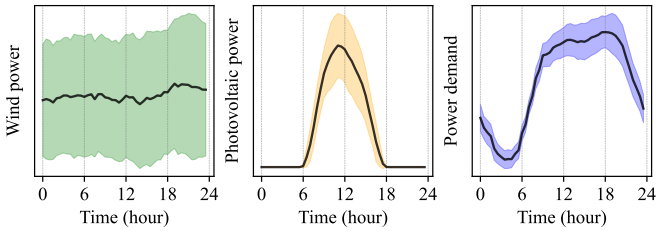


Fig. 6. Daily variation patterns of different data sources.

Second, the entropy evaluation should be adapted to the framework accordingly. Theoretically, given the action probability $p_\theta(a_t|s_t) = \prod_k^{N_{ts}} p_\theta(C_t^k|C_t^{k-1})$, the entropy is evaluated to be $-\mathbb{E}_{a_t\sim\tau_\pi}\log p_\theta(a_t|s_t)$, which may be numerically unstable due to the long probability chain. Also, it is inconvenient to calculate the theoretical maximum entropy to facilitate a more tunable entropy regularization (i.e., $\bar{H} = \log|\mathcal{A}|$ and $|\mathcal{A}|$ counts the total number of infeasible actions of DDNR). It is common to set a certain entropy ratio for discrete policy, e.g., $\frac{\mathcal{H}(\pi(\cdot|s_t))}{\bar{H}}$=0.98 [30]. Hence, we approximate the entropy ratio using the sum of entropies of each sub-action by:

$$\frac{H_t}{\bar{H}} \approx -\mathbb{E}_{a_t\sim\tau_\pi}\sum_{k=1}^{N_{ts}}\frac{\mathcal{H}\left(p_\theta(C_t^k|C_t^{k-1})\right)}{N_{ts}\cdot\log\sum_{i=1}^{|\mathcal{E}_S|}M_{t,i}^k}, \quad (22)$$

where H_t denotes the approximated entropy normalized into [0,1] using the single-step maximum entropy ($\log\sum_{i=1}^{|\mathcal{E}_S|}M_{t,i}^k$). One can maintain a certain entropy level for SAC by dynamically adjusting the weight α to balance exploration and exploitation as follows [31]:

$$\nabla_\alpha J(\alpha) = \nabla_\alpha\alpha\left(\frac{H_t}{\bar{H}} - \hat{H}\right), \quad (23)$$

where $\hat{H}$ is the target entropy ratio for the agent to track. With these adaptations, SAC can be trained stably for DDNR to generate configurations flexibly and safely.

D. Pseudo-codes of the Proposed Method

The detailed implementation of the proposed DRL method for the DDNR problem is summarized in **Algorithm**.

V. Case Study

The section examines the proposed method in various cases with different system scales and uncertainty levels and compares it with typical benchmarks and action strategies in terms of performance and behavior patterns.

A. Environmental Settings

The proposed method is first tested on the IEEE 33-bus PDN [7], which is modified to enable DDNR, as illustrated in Fig. 5. Five tie lines (i.e., $N_{ts} = 5$) equipped with tie switches are added to the network, connecting buses 25-28, 17-30, 7-22, 8-13, and 11-33. Ten lines are chosen to be equipped with sectionalizing switches, including lines 6-7, 7-8, 9-10, 11-12, 14-15, 19-20, 23-24, 26-27, 28-29, and 30-31. The PDN is assumed to operate at a low-voltage level of 15kV. For modelling the real dynamics and uncertainty in practice, the real-world power demand data is adopted to simulate power dynamics at buses from 2 to 33, referring to [25]. To tackle operation challenges in the presence of high-penetration RESs, DDNR is applied to wind power (WP)-dominated and photovoltaics (PV)-dominated PDNs, respectively, which is realized by real-world time series of WP [32] and PV [25]. The patterns of these time series are presented in Fig. 6. RESs are assumed to be installed at buses 14, 18, 21, 25, and 29, with a rated capacity of $S_i^g = 2000$ kW. Maximum loads of each bus are increased by 1.5 and 2 times for wind and solar power-dominated PDNs, respectively. For the reward, C_p and C_a are set as 0.13$/kWh and 1$/switching, respectively, referring to [3]. With voltages in p.u., C_v is set as 10 to penalize excess voltage deviations. To balance the multiple goals, k_v and k_e are set as 1 and 0.023, respectively. $r_{c,t}$ is set as 100 to penalize the actions violating (4). The decision interval is set as 1 hour, and an episode has 24 decision steps in a day. During training, the one-year time series of renewables and demands are used to simulate PDNs, and one hundred episodes simulated by the next-year data are used for evaluation. In each episode, PDNs start to operate with tie switches opened and sectionalizing switches closed [3].

B. Settings of the Proposed Method and Benchmarks

Various benchmarks are employed to demonstrate the advantages of the proposed method. Given the discrete action space, the value-based DDQN [12], [23], [16] is used for comparison with the actor-critic-based SAC [3], [33], [11]. Programming methods [7], [3] are also used to calibrate the relative performance of DRLs. They are given as follows.

SAC-S (Ours): It denotes the proposed sequential masking-based SAC algorithm.

SAC-V & DDQN-V: They sample actions from a valid action set that encodes all the feasible actions according to [13]. In this case, the total number of feasible actions is 609, compared to its original vast action space of $2^{|\mathcal{E}_s|} = 32768$ with $|\mathcal{E}_s| = 15$, leading to a rate of feasible actions of 1.86% at a decision step.

SAC-E & DDQN-E: Their action space is restricted by the rule of "exchanging the status of a pair of RCSs" with a size of $|\mathcal{E}_s|^2 - |\mathcal{E}_s| + 1 = 226$. At each step, they utilize a mask to avoid the violation of the radial constraint in Eq. (4). The total number of links via "status exchange" between any two feasible configurations amounts to 6182, leading to a rate of feasible actions of 9.43% on average at a decision step.

SAC-R: It adapts a reward-based approach to address infeasible actions. A large penalty will be returned if the constraint in Eq. (4) is violated, after which an episode is terminated. As the agent may frequently sample infeasible actions, the action space of SAC-R is encoded in the same way as SAC-E with a higher action feasibility rate, referring to [12]. Since the ϵ-greedy exploration strategy of DDQN induces frequent violations that diverge the learning process, DDQN is not adopted for comparison.

TABLE I: CRITICAL HYPERPARAMETERS FOR TRAINING DRLS

Hyper-parameter	Value	Hyper-parameter	Value
Batch size $\|\mathcal{B}\|$	32	Number of episodes N_e	1e5
Replay buffer size $\|\mathcal{D}\|$	3e5	Number of gradient steps N_s	10
Discount factor γ	0.96	Polyak averaging factor ρ	1e-3
Learning rates η_π, η_Q	1e-4	Annealing rate μ	1-5e-5
Learning rate η_α	1e-3	Decay rate of DDQN	1-5e-5
Initial entropy target $\widehat{H}$	0.98		

SSO: With the complete and accurate parameters of a PDN, the MIP model can be used to implement single-step optimization (SSO) of DDNR, referring to the model described in Appendix [7], [34]. Therefore, MIP is used to demonstrate the gaps between model-free DRL methods and the traditional model-based benchmark.

MPC: Considering the dynamic-programming nature of DDNR, we also provide the solutions produced by a model predictive control (MPC)-like benchmark, referring to [3], which optimizes multiple future decision steps at a time and just implements the solution to the current decision. Instead of using predicted parameters, our implementation directly incorporates the accurate future parameters, so as to present "almost optimal solutions" for calibrating the performance of other methods. Four future steps are considered in this case, given its computational burden.

MPC-E: It is MPC constrained by the same restricted action space as that of SAC-E, in order to present the "almost actual" performance gap incurred by the action strategy of SAC-E.

DRL methods use consistent hyper-parameters for a fair comparison, and the critical ones are provided in Table I. The DDQN agents (Q networks) and critics in all the SAC agents are built by 100×4 FCNs. For the actors in SAC agents, SAC-E and SAC-R employ 100×4 FCNs, and the proposed SAC-S encodes state features by using 100×2 FCNs and then generates sub-actions by stacking a 100×1 GRU and 100×2 FCNs. Three-step historical features ($|\mathcal{T}_{hist}| = 3$) are used for learning. FCNs use the ReLU to introduce non-linearity.

In the case study, we emphasize comparing different DRL action strategies to promote our proposed strategy for various data-driven applications based on DDNR. Comparisons between DRL and model-based methods are majorly used to provide approximate optimality gaps (e.g., MPC/MPC-E) and comparative insights (e.g., SSO). In practice, comparisons between model-based and model-free methods rely on various settings, such as the sufficiency of system observation [18], the accuracy of state estimation [3], and the costs of switching [11]. This paper assumes complete and accurate observations with proper switching costs [3] for model-based benchmarks, showing theoretical gaps for various methods. The performance and scalability comparison of model-free methods with various action strategies is of higher importance in the following subsections.

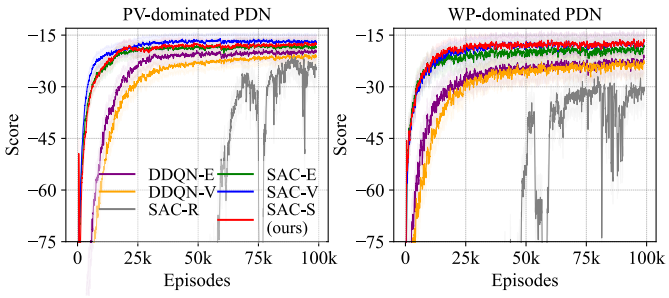


Fig. 7. Learning curves of different DRL methods on 33-bus PDNs.

TABLE II: EVALUATION ON PV-DOMINATED PDN (LOW UNCERTAINTY)

Method	P_{cost}		A_{cost}		V_{dev}	R_{viol}	*Score*	*Gap*
	(kWh)	($)	(NoT)	($)	(%)	(%)		(%)
No DDNR	2450	318.4	-	-	4.57	37.04	-57.1	248.7
MPC	1780	231.4	17.7	17.7	2.93	4.29	-16.4	0
MPC-E	1828	237.7	16.3	16.3	3.01	4.71	-17.0	3.6
SSO	1836	238.7	19.1	19.1	3.11	6.21	-18.2	10.8
DDQN-E	1894	246.2	21.6	21.6	3.26	6.67	-19.1	16.8
DDQN-V	1941	252.3	75.9	75.9	3.25	5.33	-20.1	22.9
SAC-R	2218	288.4	12.2	12.2	3.36	12.58	-23.0	40.3
SAC-E	1806	234.7	12.3	12.3	3.11	4.83	-17.4	6.1
SAC-V	1793	233.2	20.4	20.4	3.00	4.46	-16.9	3.0
SAC-S (Ours)	1800	234.1	26.0	26.0	3.01	4.46	-17.1	4.5

Note: *Score* is the average accumulated reward $\sum_{t\in\mathcal{T}} r_t$, NoT is the number of times of switching, R_{viol} is the average rate of voltage violation, *Gap* is the performance gap with the best *Score*.

TABLE III: EVALUATION ON WP-DOMINATED PDN (HIGH UNCERTAINTY)

Method	P_{cost}		A_{cost}		V_{dev}	R_{viol}	*Score*	*Gap*
	(kWh)	($)	(NoT)	($)	(%)	(%)		(%)
No DDNR	3523	458.0	-	-	4.73	40.54	-65.3	393.9
MPC	2224	289.2	28.7	28.7	2.10	2.12	-13.2	0
MPC-E	2290	297.7	23.8	23.8	2.21	3.33	-14.4	8.7
SSO	2344	304.7	30.9	30.9	2.48	3.96	-15.9	20.6
DDQN-E	2613	339.8	41.3	41.3	2.83	8.38	-20.6	55.9
DDQN-V	2680	348.4	78.0	78.0	2.85	6.88	-21.0	58.7
SAC-R	2688	349.5	5.6	5.6	3.10	11.62	-23.7	79.4
SAC-E	2464	320.3	15.3	15.3	2.68	6.21	-18.0	36.4
SAC-V	2383	309.7	31.2	31.2	2.57	4.33	-16.4	24.0
SAC-S (Ours)	2323	302.0	39.4	39.4	2.55	3.88	-16.3	23.1

C. *Learning Curves of DRL Methods*

The learning curves of DRL methods are illustrated in Fig. 7. The major observations are given as follows. 1) The learning curves of the PV-dominated PDN feature noticeably lower variance, and their *Scores* (accumulated rewards) are closer than that of the WP-dominated PDN due to much lower uncertainty of PV as shown in Fig. 6. 2) Most DRLs can converge stably except SAC-R, demonstrating that frequent large penalty is prone to result in learning inefficiency and hinder performance. 3) SACs learn better than DDQNs likely due to their policy-based exploration strategy and entropy regularization. 4) SAC-S presents a close learning curve to the best-performing SAC-V, showing that it allows the agent to reach any feasible configuration freely like SAC-V. Also, SAC-S can outperform SAC-E to prove that the action space reduction does restrict optimality.

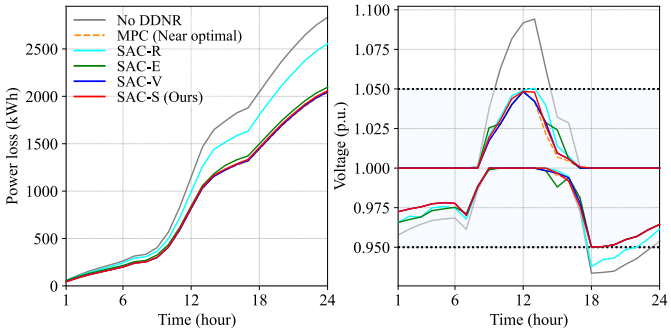


Fig. 8. Typical operation patterns of the PV-dominated PDN.

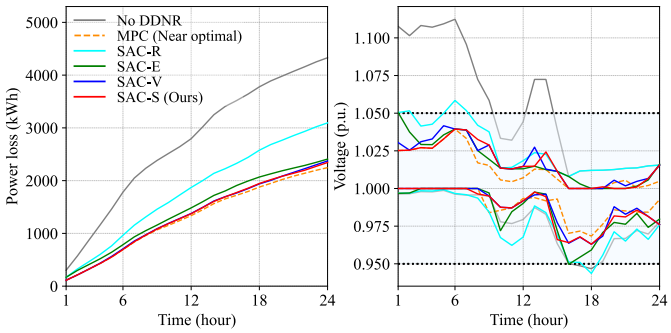


Fig. 9. Typical operation patterns of the WP-dominated PDN.

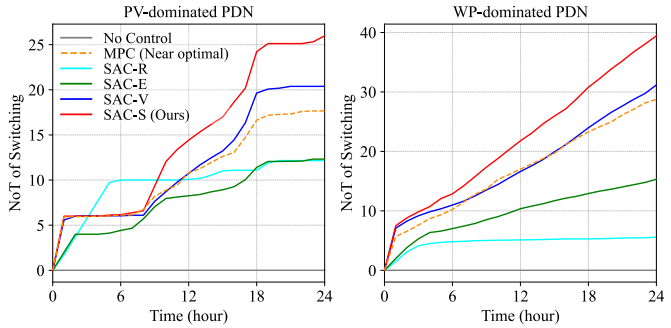


Fig. 10. Daily number of times (NoT) of switching on average

D. *Comparative Analysis of Different DDNR Methods*

The evaluation results averaged over one hundred episodes are presented in Tables II and III. Regulation dynamics are visualized in Figs. 8 and 9. Without DDNR, the PV-dominated PDN has a severe voltage violation rate due to the concurrent PV generation from 10 am to 2 pm and heavy power demand after 8 pm, as illustrated in Fig. 8. Likewise, concurrent WP generation can also insecure PDN and cause over-voltage issues. The difference is that the timings of concurrent WP generation are irregular and hardly predictable compared to PV generation. For instance, in Fig. 9, WP can even lead to voltage-violated durations twice a day, i.e., before 6 am and around 2 pm. Such an uncertainty of WP inevitably makes the problem challenging.

It is undoubtful that PDNs with an effective DDNR can operate more securely and economically. Among these methods, MPC incorporates accurate future parameters to eliminate uncertainty to outperform others easily. Calibrated by the *Score* of MPC, the *Gap* of *Scores* is used to evaluate the relative performance of other methods, as shown in Tables II and III. We offer the following conclusions in terms of comprehensive

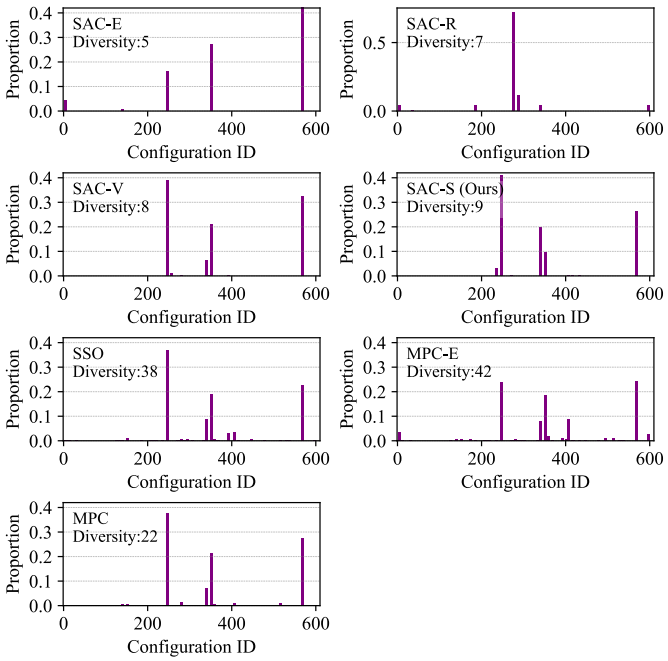


Fig. 11. Distribution of visited configurations on PV-dominated PDN.

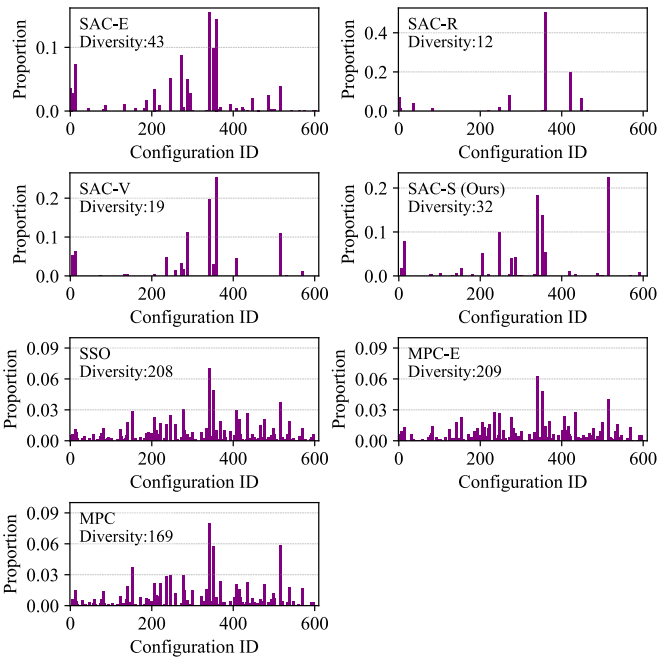


Fig. 12. Distribution of visited configurations on WP-dominated PDN.

performance. 1) The increased uncertainty of WP causes a considerable *Gap* (over 20%) between MPC and the remaining methods due to larger prediction errors than in the cases with PV generation (they have smaller *Gaps* less than 10%). 2) SAC-V and SAC-S have similar *Scores*, verifying the effectiveness of the sequential masking strategy to address the DDNR action space. 3) SAC-S/V can perform better than SAC-E consistently on both PDNs (e.g., the WP case shows a *Gap* of around 10%), proving that the reduced action space of SAC-E does restrict performance. 4) The *Gap* incurred by the "-E" constraint can also be reflected by comparing MPC and MPC-E, which is enlarged by an increased uncertainty from 3.6% with PV to 8.7% with WP. MPC-E outperforms SAC-S/V in the WP case, reflecting the potential of SAC-E. However, the larger uncertainty entails frequent configuration changes and reduces predictability for dynamics, making it more challenging for SAC-E to plan for a configuration in advance. 5) DDQNs deliver poorer policies than their SAC counterparts, and interestingly, DDQN cannot handle the valid action set sufficiently, leading to a poorer *Score* of DDQN-V than that of DDQN-E in both cases. DDQN-V with the ϵ -greedy exploration may have a higher possibility to sample unfavorable configurations than DDQN-E to reduce its learning efficiency, as indicated by the lower curves of DDQN-V throughout training in Fig. 7. 6) SAC-S is 5% better than the model-based SSO in the PV case, whereas they have similar results in the WP case. It shows that DRLs can be more advantageous when PDN dynamics are more predictable, so that DRLs can make more preventive decisions considering future scenarios. Note that a higher action cost can further enlarge gaps between SSO and DRLs [11]. 7) SAC-R cannot deliver sound solutions to DDNR with frequent penalties.

As for voltage regulation (a security range between 0.95 p.u. and 1.05 p.u.), PDNs suffer voltage violations in around 40% of operation time without DDNR. In contrast, an effective DDNR can significantly reduce the violation rate to less than 5%. Average voltage deviations (V_{dec}) are generally consistent with their *Scores*. In the PV-dominated case, the voltage profiles are quite similar due to less uncertainty, as shown in Fig. 8. In contrast, they have distinct voltage dynamics in the WP-dominated case. For instance, SAC-E, which cannot switch to an ideal configuration in one step, is likely to infringe the security code (e.g., at 1 am and 4 pm in Fig. 9). In contrast, SAC-S, closely approximating SAC-V, has lower risk of violations to show that its action strategy can definitely benefit DDNR problems.

DDNR achieves around 50$ and 100$ savings ($P_{cost} + A_{cost}$) per day in PV and WP-dominated cases, respectively. There is no significant difference in power loss optimization among these methods except SAC-R, as power losses are less sensitive to topological changes. On the contrary, they present inconsistent NoTs of switching due to adopting different action strategies as will be analyzed in Section IV-E.

E. Action Pattern Comparison and Analysis

The NoTs of switching are illustrated in Fig. 10. We give the following observations: 1) The NoT of switching grows fast when dynamics change significantly. Switching actions increase fast in two stages in the PV case, i.e., the initial stage with a default configuration at 0 am and the stage of varying PV and demand between 8 am and 7 pm. In the WP case, however, WP varies intermittently throughout the day, leading to evenly implemented switching actions after the initial stage. 2) Owing to the action space reduction, SAC-E and SAC-R act less frequently than others in both cases, restricting their optimality. 3) Exploring in the same action space, SAC-S generally acts more frequently than SAC-V. The increased switching may result from the recurrent calculation of sub-actions in SAC-S, where a long probability chain may preferably generate more diverse configurations between successive states than SAC-V.

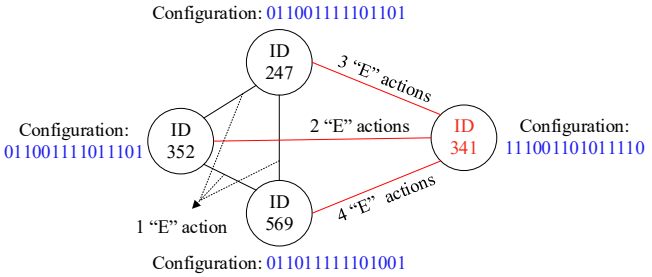


Fig. 13. Number of actions for SAC-E to jump between four major configurations exploited by MPC in the PV-dominated case.

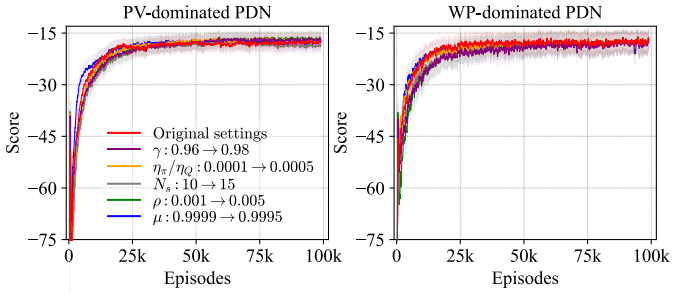


Fig. 14. Training stability of our method with different hyperparameters.

Nevertheless, SAC-S can still deliver competitive results like SAC-V but has better scalability than SAC-V.

The distributions of visited configurations are illustrated in Figs. 11 and 12, where the x-axis is the configuration number (a total of 609 in the cases) in an arbitrary order, and diversity means how many different configurations are visited. In the PV-dominated case, the configuration diversity is low since the PDN has regular dynamic patterns. Four critical configurations (ID and the on/off status of fifteen RCSs) identified by MPC are 247 (011001111101101), 341 (111001101011110), 352 (011001111011101), and 569 (011011111101001) in Fig. 11. It can be found that SAC-E fails to identify configuration 341 in Fig. 11, which requires at least two decision steps (i.e., the "E" action) for SAC-E to reach it from the other three configurations, as illustrated by Fig. 13. It reveals the difficulty for SAC-E to learn the optimal solutions. The same may apply to a harder task in Fig. 12, where optimal solutions frequently jump among distinct configurations due to fast-varying states, making it more challenging for SAC-E to approach optimality. The comparison between MPC and MPC-E reveals that though MPC-E can effectively identify the configuration 341 missed by SAC-E, its visitation distribution is proportionally different from that of MPC due to the "-E" constraint due to using more intermediate configurations to reach configuration 341. In contrast, SAC-S and SAC-V capably identify the four configurations, though their proportions may slightly deviate from that of MPC. Tasks become challenging in WP cases, as stochastic WP generation produces various operation dynamics. It can be reflected by the significantly increased diversity of MPC actions. Due to the dimensionality curse, it is always challenging for DRLs with finite sampling times to search for the best configurations. Hence, DRLs tend to exploit tens of configurations to tackle the complexity, whereas the model-based methods can directly calculate the best solution to their linearized model. Interestingly, SAC-E presents a greater diversity than SAC-V and SAC-S, likely because SAC-E has to visit more temporary configurations to reach expected configurations in the fast-varying dynamics.

F. Hyperparameter Sensitivity and Training Stability

DRL is often sensitive to hyperparameters and may converge unstably during training. This section presents the hyperparameter sensitivity of our method (SAC-S) by comparing it against its strong competitor, SAC-E, and illustrates its learning stability under different hyperparameter settings. Five critical hyperparameters are selected from Table I, i.e., the discount factor γ, learning rate η_π/η_Q, number of gradient steps N_s, averaging factor ρ, and annealing rate μ. Each one is moderately modified to form a new set of hyperparameters for evaluation.

TABLE IV: HYPERPARAMETER SENSITIVITY EVALUATION ON PV- AND WP-DOMINATED PDNS

Hyperparameters	Explanation	PV-domin. PDN (low uncertainty)		WP-domin. PDN (high uncertainty)	
		Ours	SAC-E	Ours	SAC-E
Original settings	-	-17.1	-17.4	-16.3	-18.0
γ: 0.96 → 0.98	Longer sighted	-16.9	-17.6	-16.6	-17.8
η_π/η_Q: 1e-4 → 5e-4	Larger grad. step	-17.0	-17.2	-16.5	-17.8
N_s: 10 → 15	More grad. steps	-17.7	-17.8	-16.3	-18.1
ρ: 1e-3 → 5e-3	Less delayed critic	-16.9	-17.3	-16.3	-17.7
μ: 1-1e-4 → 1-5e-4	Less exploration	-17.3	-18.0	-16.5	-18.8
Average *Score*		-17.2	-17.6	-16.4	-18.0
Average *Gap* (%)		0	2.3	0	9.8

The training curves of SAC-S are presented in Fig. 14, and numerical results compared with SAC-E are given in Table IV. The converged training curves in Fig. 14 have small variances after sufficient training rounds, showcasing that our method can learn efficiently and stably for the DDNR problem with proper hyperparameters and under typical reward shaping [3], [25]. Table IV validates that the evaluation performance of SAC-S is less sensitive to proper hyperparameters, though reducing the exploration rate (μ) unsurprisingly compromises the performance of both SAC-S and SAC-E. Results reveal that SAC-S can consistently outperform the SAC-E even in the low-uncertainty case (i.e., a small but valid average gap of 2.3%), and the restricted action space does limit the performance of SAC-E, especially when facing high uncertainty in PDNs (i.e., an average gap of 9.8%).

G. Evaluation on Large Power Distribution Network

We also evaluate the proposed method on a larger IEEE 118-bus PDN [3], as demonstrated in Fig. 15, to present the scalability of the proposed method. In this case, fifteen tie lines (red dotted lines) are assumed (i.e., $N_{ts} = 15$), and twenty normally connected lines (red solid lines) are selected for DDNR. The system is assumed to operate at 10 kV. Both PV and WP generation are considered in the PDN. Six PV and WP plants with a rated capacity S_i^g of 3000kW are assumed to be distributed in the PDN. As power loss increases by around three times, k_e is set as one-third of the one used in the 33-bus PDN, and C_a is assumed to be 2$/switching. Hyperparameters are consistent as before, except that hidden layers are increased from 100 to 200 for every agent, and MPC solves three-step decisions considering the increased problem scale.

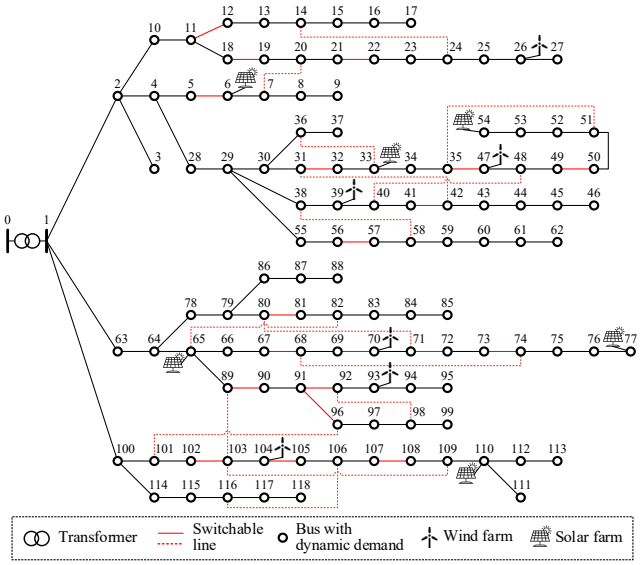


Fig. 15. A modified IEEE 118-bus PDN for the DDNR problem.

TABLE V: EVALUATION ON 118-BUS PDN WITH MIXED RENEWABLES

Method	P_{cost}		A_{cost}		V_{dev}	R_{viol}	*Score*	*Gap*
	(kWh)	($)	(NoT)	($)	(%)	(%)		(%)
No DDNR	11371	1478.3	-	-	5.85	61.92	-93.5	471.5
MPC	7260	943.8	45.0	89.9	2.71	2.87	-16.4	0
MPC-E	7428	965.6	35.8	71.6	2.86	2.92	-17.2	5.1
SSO	7641	993.3	42.5	85.1	3.03	4.21	-18.8	15.1
SAC-R	8144	1058.8	10.4	20.8	3.90	20.08	-33.4	104.1
SAC-E	7551	981.6	28.0	56.0	3.09	3.58	-18.6	13.5
SAC-V	-	-	-	-	-	-	-	-
SAC-S (Ours)	7434	966.5	68.2	136.4	2.92	3.00	-18.0	10.1

With such a setting, even with an action feasibility rate of 0.01%, the number of feasible configurations can be more than one million ($2^{35} \times 0.01\%$), making SAC-V inapplicable to the problem. The evaluation results averaging over one hundred episodes are given in Table V. SAC-R can learn even worse policy on the large-scale PDN with a *Gap* of more than 100%. In the problem scale, while SAC-E is more computationally scalable than SAC-S (heavy calculation of RNN recurrence and masking operations), SAC-E is more inflexible in the aspect that it has to take more steps to reach an optimal configuration (the number of steps is fifteen at most between two configurations compared to just five in the 33-bus case). Combining these pros and cons, SAC-S can still deliver a better result than SAC-E and SSO, showing its greater potential to benefit PDN operators as a data-driven and model-free alternative to traditional methods. Note that RNNs may suffer a gradient vanishment problem once the recurrent gradient path is too long. Hence, there is definitely some room for performance enhancement using more advanced neural structures, such as attention mechanisms [35]. The proposed strategy can be compatible with such improvements, inspiring others to strengthen data-driven solutions.

H. Comparison with A Different Sub-Action Strategy

Another feasible way to implement our learning framework is to use "exchanging the status of a pair of RCSs" as a sub-action since it takes at most N_{ts} steps to reach an arbitrary configuration. Two major modifications are made to adapt our learning framework to the sub-action strategy: 1) Since sub-actions have a variable number of decision steps, we add a "Stop" option in the sub-action space for DRL to make no configuration change and stop selecting the next sub-action. Also, the number of sub-actions will be capped by N_{ts}. 2) As these new sub-actions have no additive relation in Eq. (19), we concatenate and feed them into the critic network. For those numbers of sub-actions less than N_{ts}, we pad the rest of the action vector with one-hot vectors that select the "Stop" option.

TABLE VI: COMPARISON WITH A DIFFERENT SUB-ACTION STRATEGY

Method	P_{cost}		A_{cost}		V_{dev}	R_{viol}	*Score*	*Gap*
	(kWh)	($)	(NoT)	($)	(%)	(%)		(%)
PV-Dominated PDN (low uncertainty)								
SAC-S (Ours)	1800	234.1	26.0	26.0	3.01	4.46	-17.1	0
SAC-SE-Stat.	1802	234.2	20.6	20.6	3.05	4.63	-17.2	0.9
SAC-SE-Dyn.	1911	248.5	14.2	14.2	3.11	6.25	-17.9	4.7
WP-Dominated PDN (high uncertainty)								
SAC-S (Ours)	2323	302.0	39.4	39.4	2.55	3.88	-16.3	0
SAC-SE-Stat.	2407	312.9	22.5	22.5	2.62	5.04	-16.7	2.8
SAC-SE-Dyn.	2468	320.8	16.0	16.0	2.63	5.46	-17.2	5.5

There are two different ways for DRL to learn the new sub-action strategy. A natural choice is to let DRL generate sub-actions starting from the current configuration C_t, which is **dynamic** in the learning process. In contrast, it is also possible to always start with a **static** initial configuration C_0 since the PDN only implements the final configuration constructed by all the sub-actions. The latter one is similar to the proposed SAC-S. Hence, two benchmarks based on the sequential-masking learning framework and "exchanging-status" sub-actions are named **SAC-SE-Dynamic** and **SAC-SE-Static** for comparison.

The numerical results of cases with different uncertainty are presented in Table VI. It can be found that: 1) SAC-SE-Dynamic delivers poorer *Scores* in both cases, revealing that dynamic initial configurations may lead to higher learning difficulty for DRL. A reason may be that DRL has to learn more transitions from a given configuration C_t to a target configuration C_{t+1}. Specifically, if there are n configurations to be frequently visited, there will be at most $n(n-1)$ transitions of configurations for DRL to explore, in order to jump between any two of these configurations. In contrast, DRL can only exploit at least n transitions from a static initial configuration C_0 to the target configuration C_t. 2) SAC-SE-Static shows a close performance to SAC-S, showing the potential of integrating other sub-action strategies into our learning framework. SAC-SE-Static falls slightly behind SAC-S in the high-uncertainty case, probably due to their nuances in the sub-action behavior (e.g., an extra "Stop" option) and sub-action space (e.g., different dimension and sparsity). 3) Their *Scores* are noticeably better than the Score of SAC-E (i.e., -18.0) in the high-uncertainty case, demonstrating the effectiveness of action space decomposition in tackling the complex action space of DDNR. 4) SAC-SEs present less action cost than SAC-S, as the "Stop" option of SAC-SEs may result in more actions with fewer NoT of switching than SAC-S during exploration.

VI. Conclusion

This paper proposes a sequential masking strategy to effectively address the vast and constrained action space of the DDNR problem. A GRU-based SAC algorithm with scalable and data-efficient features is designed for the proposed strategy. Three types of typical DRL methods for DDNR are reviewed and compared to demonstrate the motivation and advantage of the proposed strategy. Comparative case studies are conducted with various benchmarks, and the results conclude that:

1) Compared with other DRL benchmarks, the proposed method (SAC-S) can

- significantly outperform SAC-R with a gap of 34.5%, 45.4%, and 85.6% in cases with different uncertainties and system scales, showing that "penalizing infeasible action in the reward" is not a proper solution for DDNR.
- consistently outperform SAC-E in various cases with a large gap of 10.4% in the high-uncertainty case while a small gap of only 1.7% in the low-uncertainty case, revealing that SAC-E with a restricted action space can only be a good choice for scenarios where system dynamics vary slowly and deterministically.
- be more scalable than SAC-R in the large-system case despite similar performance in small-system cases, as the neural agent is highly inefficient in training with millions of feasible configurations as its output layer.

2) DRLs can achieve a minimum optimality gap of 4.5% with the approximated optimum (MPC) in the low-uncertainty case, in which the proposed method can surpass the model-based SSO. In contrast, the gap is larger with 23.1% in the high-uncertainty case, showing that uncertainty is a key factor to impact relative performance of DRL in DDNR.

3) DRLs tend to exploit around 5 to 10 times less diverse configurations than model-based approaches, especially in the high-uncertainty case. This reflects a generalization challenge of DRLs trained by finite sampling times in handling a vast combinatorial action space.

The proposed method further strengthens the data-driven solution to DDNR, serving as a valuable model-free option to operate PDNs more securely and economically.

Finally, it is worth noting that in this study GRU is used to build the agent due to its structural suitability for modeling the probability chain of decomposed sub-actions and its computational simplicity compared with LSTM and attention modeling. This allows us to focus on the actual performance enhanced by the proposed action strategy. Besides, SAC is adopted for algorithmic optimization thanks to its higher data efficiency as an off-policy algorithm than on-policy counterparts like REINFORCE and PPO and its greater performance than Q-learning algorithms like DDQN. In the future, one may equip the proposed strategy with more advanced NN structures like graph NNs (GNNs) and Transformers for further improvement. Moreover, some DRL techniques like imitation learning that incorporates expert knowledge or advanced exploration strategy may also be instrumental in searching for promising solutions to tackle the challenge posed by the constrained and vast action space of DDNR.

Appendix

The mathematical formulation of the DDNR problem for MIP is given as follows.

$$\min_{\{b_{ij,t}|ij\in\mathcal{E}_s,t\in\mathcal{T}\}} \sum_{t\in\mathcal{T}} k_v V_t^{dev} + k_e\left(C_p P_t^L + C_a A_t\right) \tag{24}$$

subject to

$$\begin{aligned} V_{i,t} - V_{j,t} &\le \left(1-b_{ij,t}\right)M + R_{ij}p_{ij,t} + X_{ij}q_{ij,t}, \\ &\forall i,j\in\mathcal{N},\ \forall ij\in\mathcal{E} \\ V_{i,t} - V_{j,t} &\ge \left(b_{ij,t}-1\right)M + R_{ij}p_{ij,t} + X_{ij}q_{ij,t}, \\ &\forall i,j\in\mathcal{N},\ \forall ij\in\mathcal{E} \end{aligned} \tag{24a}$$

$$\begin{aligned} \textstyle\sum_{j\in\Omega(i)} p_{ij,t} &= P_{i,t}^g - P_{i,t}^d,\ \forall i\in\mathcal{N} \\ \textstyle\sum_{j\in\Omega(i)} q_{ij,t} &= Q_{i,t}^g - Q_{i,t}^d,\ \forall i\in\mathcal{N} \end{aligned} \tag{24b}$$

$$\begin{aligned} \beta_{ij,t} + \beta_{ji,t} &= b_{ij,t},\ \forall ij\in\mathcal{E} \\ \textstyle\sum_{j\in\Omega(i)} \beta_{ij,t} &= 1,\ \forall i\in\mathcal{N} \\ \beta_{0j,t} &= 0,\ \forall j\in\Omega(0) \end{aligned} \tag{24c}$$

$$\begin{aligned} V_t^{dev} &= \left(\frac{V_t^m}{V_B}\right)^2 \\ V_t^m &\ge V_{i,t} - 1,\ \forall i\in\mathcal{N} \\ V_t^m &\ge 1 - V_{i,t},\ \forall i\in\mathcal{N} \end{aligned} \tag{24d}$$

$$P_t^L = \textstyle\sum_{ij\in\mathcal{E}} R_{ij}\left(p_{ij,t}^2 + q_{ij,t}^2\right) \tag{24e}$$

$$\begin{aligned} A_t &= \textstyle\sum_{ij\in\mathcal{E}} \alpha_{ij,t} \\ \alpha_{ij,t} &\ge b_{ij,t} - b_{ij,t-1},\ \forall ij\in\mathcal{E} \\ \alpha_{ij,t} &\ge b_{ij,t-1} - b_{ij,t},\ \forall ij\in\mathcal{E} \end{aligned} \tag{24f}$$

Constraints (24a) are the linearized DistFlow model [34] that approximates line flow equations in Eq. (1), where a big M constant is used to introduce the binary switching variable $b_{ij,t}$, and R_{ij} and X_{ij} are line resistance and reactance, respectively. Constraints (24b) are power balance equations at each node. Constraints (24c) are the spanning tree constraints [7] to guarantee a radial topology of PDN, where $\Omega(i)$ is a set of adjacent nodes to the node i, $\beta_{ij,t}$ is the binary variable, and $i = 0$ means the root node (a substation) of a PDN. Constraints (24d) define the voltage penalty V_t^{dev} for the maximum voltage deviation V_t^m, normalized by a security margin V_B. The constraint (24e) defines the total power loss P_t^L. The constraints (24f) define the total NoT of switching A_t, where $\alpha_{ij,t}$ counts the status change of an RCS.

For the consistency with the bowl-shaped voltage reward in Eq. (7), V_t^{dev} is then modified as follows.

$$\begin{aligned} V_t^s &\ge V_t^m - \zeta M \\ V_t^s &\ge V_B\zeta \\ V_t^v &\ge V_t^m - V_B \\ V_t^v &\ge 0 \\ V_t^{dev} &= \left(\frac{V_t^s}{V_B}\right)^2 + C_v\frac{V_t^v}{V_B} \end{aligned} \tag{25}$$

where ζ is a binary variable, and variables V_t^s and V_t^v are defined as safe and violated voltage deviations, respectively.

REFERENCES

[1] M. J. Ghadi, S. Ghavidel, A. Rajabi, A. Azizivahed, L. Li, and J. Zhang, "A review on economic and technical operation of active distribution systems," *Renew. Sust. Energ. Rev.,* vol. 104, pp. 38-53, 2019.

[2] N. Liu, C. Li, L. Chen, and J. Wang, "Hybrid data-driven and model-based distribution network reconfiguration with lossless model reduction," *IEEE Trans. Ind. Inf.,* vol. 18, no. 5, pp. 2943-2954, 2021.

[3] Y. Gao, W. Wang, J. Shi, and N. Yu, "Batch-constrained reinforcement learning for dynamic distribution network reconfiguration," *IEEE Trans. Smart Grid,* vol. 11, no. 6, pp. 5357-5369, 2020.

[4] Y. Li, G. Hao, Y. Liu, Y. Yu, Z. Ni, and Y. Zhao, "Many-objective distribution network reconfiguration via deep reinforcement learning assisted optimization algorithm," *IEEE Trans. Power Deliv.,* vol. 37, no. 3, pp. 2230-2244, 2021.

[5] W. Huang, W. Zheng, and D. J. Hill, "Distribution network reconfiguration for short-term voltage stability enhancement: An efficient deep learning approach," *IEEE Trans. Smart Grid,* vol. 12, no. 6, pp. 5385-5395, 2021.

[6] B. Sultana, M. Mustafa, U. Sultana, and A. R. Bhatti, "Review on reliability improvement and power loss reduction in distribution system via network reconfiguration," *Renew. Sust. Energ. Rev.,* vol. 66, pp. 297-310, 2016.

[7] S. Lei, Y. Hou, F. Qiu, and J. Yan, "Identification of critical switches for integrating renewable distributed generation by dynamic network reconfiguration," *IEEE Trans. Sustain. Energy,* vol. 9, no. 1, pp. 420-432, 2017.

[8] R. Jabr, R. Singh, and B. Pal, "Minimum loss network reconfiguration using mixed-integer convex programming," *IEEE Trans. Power Syst.,* vol. 27, no. 2, pp. 1106-1115, 2012.

[9] O. Badran, S. Mekhilef, H. Mokhlis, and W. Dahalan, "Optimal reconfiguration of distribution system connected with distributed generations: A review of different methodologies," *Renew. Sust. Energ. Rev.,* vol. 73, pp. 854-867, 2017.

[10] R. S. Rao, K. Ravindra, K. Satish, and S. Narasimham, "Power loss minimization in distribution system using network reconfiguration in the presence of distributed generation," *IEEE Trans. Power Syst.,* vol. 28, no. 1, pp. 317-325, 2012.

[11] R. Wang, X. Bi, and S. Bu, "Real-Time Coordination of Dynamic Network Reconfiguration and Volt-VAR Control in Active Distribution Network: A Graph-Aware Deep Reinforcement Learning Approach," *IEEE Trans. Smart Grid,* 2023.

[12] S. H. Oh, Y. T. Yoon, and S. W. Kim, "Online reconfiguration scheme of self-sufficient distribution network based on a reinforcement learning approach," *Appl. Energy,* vol. 280, p. 115900, 2020.

[13] B. Wang, H. Zhu, H. Xu, Y. Bao, and H. Di, "Distribution network reconfiguration based on NoisyNet deep Q-learning network," *IEEE Access,* vol. 9, pp. 90358-90365, 2021.

[14] C. Wang, S. Lei, P. Ju, C. Chen, C. Peng, and Y. Hou, "MDP-based distribution network reconfiguration with renewable distributed generation: Approximate dynamic programming approach," *IEEE Trans. Smart Grid,* vol. 11, no. 4, pp. 3620-3631, 2020.

[15] M. R. Dorostkar-Ghamsari, M. Fotuhi-Firuzabad, M. Lehtonen, and A. Safdarian, "Value of distribution network reconfiguration in presence of renewable energy resources," *IEEE Trans. Power Syst.,* vol. 31, no. 3, pp. 1879-1888, 2015.

[16] T. Zhao and J. Wang, "Learning sequential distribution system restoration via graph-reinforcement learning," *IEEE Trans. Power Syst.,* vol. 37, no. 2, pp. 1601-1611, 2021.

[17] S. Malekshah, A. Rasouli, Y. Malekshah, A. Ramezani, and A. Malekshah, "Reliability-driven distribution power network dynamic reconfiguration in presence of distributed generation by the deep reinforcement learning method," *Alexandria Eng. J.,* vol. 61, no. 8, pp. 6541-6556, 2022.

[18] H. Gao, S. Jiang, Z. Li, R. Wang, Y. Liu, and J. Liu, "A Two-stage Multi-agent Deep Reinforcement Learning Method for Urban Distribution Network Reconfiguration Considering Switch Contribution," *IEEE Trans. Power Syst.,* 2024.

[19] N. Gholizadeh, N. Kazemi, and P. Musilek, "A Comparative Study of Reinforcement Learning Algorithms for Distribution Network Reconfiguration With Deep Q-Learning-Based Action Sampling," *IEEE Access,* vol. 11, pp. 13714-13723, 2023.

[20] O. B. Kundačina, P. M. Vidović, and M. R. Petković, "Solving dynamic distribution network reconfiguration using deep reinforcement learning," *Electr. Eng.,* pp. 1-15, 2022.

[21] S. Jo, J.-Y. Oh, Y. T. Yoon, and Y. G. Jin, "Self-healing radial distribution network reconfiguration based on deep reinforcement learning," *Results in Engineering,* vol. 22, p. 102026, 2024.

[22] B. Zhao, X. Han, Y. Ma, and Z. Li, "Fast Reconfiguration of Distribution Network Based on Deep Reinforcement Learning Algorithm," *IOP Conf. Ser. Earth Environ. Sci* vol. 571, no. 1, p. 012023, 2020.

[23] P. Xu, B. Wang, Y. Zhang, and H. Zhu, "Online topology-based voltage regulation: A computational performance enhanced algorithm based on deep reinforcement learning," *IET Gener. Transm. Distrib.,* vol. 16, no. 24, pp. 4879-4892, 2022.

[24] L. Thurner *et al.*, "pandapower—an open-source python tool for convenient modeling, analysis, and optimization of electric power systems," *IEEE Trans. Power Syst.,* vol. 33, no. 6, pp. 6510-6521, 2018.

[25] J. Wang, W. Xu, Y. Gu, W. Song, and T. C. Green, "Multi-agent reinforcement learning for active voltage control on power distribution networks," *Adv. Neural Inf. Process. Syst.,* vol. 34, pp. 3271-3284, 2021.

[26] K. Paton, "An algorithm for finding a fundamental set of cycles of a graph," *Commun. ACM,* vol. 12, no. 9, pp. 514-518, 1969.

[27] O. Vinyals, M. Fortunato, and N. Jaitly, "Pointer networks," *Adv. Neural Inf. Process. Syst.,* vol. 28, 2015.

[28] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," *Int. Conf. Mach. Learn.,* pp. 1861-1870, 2018.

[29] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with gumbel-softmax," *Int. Conf. Learn. Represent.,* 2016.

[30] P. Christodoulou, "Soft actor-critic for discrete action settings," *arXiv preprint arXiv:1910.07207,* 2019. [Online]. Available: https://arxiv.org/abs/1910.07207.

[31] T. Haarnoja *et al.*, "Soft actor-critic algorithms and applications," *arXiv preprint arXiv:1812.05905,* 2018. [Online]. Available: https://arxiv.org/abs/1812.05905.

[32] R. Wang, C. Li, W. Fu, and G. Tang, "Deep learning method based on gated recurrent unit and variational mode decomposition for short-term wind power interval prediction," *IEEE Trans. Neural Netw. Learn. Syst.,* vol. 31, no. 10, pp. 3814-3827, 2019.

[33] T. Wu, J. Wang, X. Lu, and Y. Du, "AC/DC hybrid distribution network reconfiguration with microgrid formation using multi-agent soft actor-critic," *Appl. Energy,* vol. 307, p. 118189, 2022.

[34] Y. Xu, Z. Y. Dong, R. Zhang, and D. J. Hill, "Multi-timescale coordinated voltage/var control of high renewable-penetrated distribution systems," *IEEE Trans. Power Syst.,* vol. 32, no. 6, pp. 4398-4408, 2017.

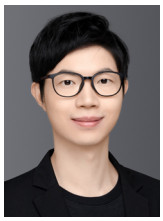
[35] M. Nazari, A. Oroojlooy, L. Snyder, and M. Takác, "Reinforcement learning for solving the vehicle routing problem," *Adv. Neural Inf. Process. Syst.,* vol. 31, 2018.

Ruoheng Wang received the B.E. degree from Wuhan Institute of Technology in 2016, the M.E. degree from Huazhong University of Science and Technology in 2018, and the Ph.D. degree from The Hong Kong Polytechnic University in 2024. He is currently a postdoctoral fellow with Department of Electrical and Electronic Engineering, The Hong Kong Polytechnic University. His current research focuses on data-driven modelling, control, optimization, and forecast of power systems considering renewable integration and transportation electrification.

Xiaowen Bi received his B.S. degree from Shandong University in 2014, and completed his M.S. and Ph.D. degrees at City University of Hong Kong in 2015 and 2020, respectively. He was a postdoctoral fellow at both The Hong Kong Polytechnic University and City University of Hong Kong. He is currently an Assistant Professor in Guangdong Provincial/Zhuhai Key Laboratory of IRADS, and Department of Statistics and Data Science, Beijing Normal-Hong Kong Baptist University. His research focuses on computational intelligence, complex networks, and their applications in intelligent transportation systems.

Siqi Bu (S'11-M'12-SM'17) received the Ph.D. degree from the electric power and energy research cluster, The Queen's University of Belfast, Belfast, U.K., where he continued his postdoctoral research work before entering industry. Then he was with National Grid UK as an experienced UK National Transmission System Planner and Operator. He is a Professor and Associate Head with Department of Electrical and Electronic Engineering, The Hong Kong Polytechnic University, Kowloon, Hong Kong, Associate Director of Research Centre for Grid Modernisation, and a Chartered Engineer with UK Royal Engineering Council, London, U.K.. His research interests include power system stability, operation and economics considering renewable energy integration, smart grid application and transport electrification.

Dr Bu is an Editor of IEEE Transactions on Power Systems, IEEE Transactions on Consumer Electronics, IEEE Power Engineering Letters, IEEE Access, IEEE Open Access Journal of Power and Energy, CSEE Journal of Power and Energy Systems, Protection and Control of Modern Power Systems, Journal of Modern Power Systems and Clean Energy, and Advances in Applied Energy. He a Fellow of IET, Chairman of IET HK Power and Energy Section, Co-Chairman of IET DPSP 2025 and APSCOM 2025, and Technical Chairman of IEEE PESIM 2026.



Zhixian Tang received the B.Eng. degree in traffic engineering from Tongji University in 2019 and the M.Eng. degree in traffic engineering from Nagoya University in 2021. He is currently a Ph.D. candidate in the Department of Electrical and Electronic Engineering at The Hong Kong Polytechnic University. His research interests include traffic signal optimization and intelligent vehicle systems.