Real-Time Power Scheduling through Reinforcement Learning from Demonstrations

Shaohuai Liu[1], Jinbo Liu[2,4], Nan Yang[3], Yupeng Huang[3], Qirong Jiang[4], Yang Gao[1,*]

***Abstract*—Real-time decision-making in power system scheduling is imperative in response to the increasing integration of renewable energy. This paper proposes a novel framework leveraging Reinforcement Learning from Demonstration (RLfD) to address complex unit commitment (UC) and optimal power flow (OPF) challenges, called GridZero-Imitation (GZ-I). Unlike traditional RL approaches that require complex reward function designs and have limited performance insurance, our method employs intuitive rewards and expert demonstrations to regularize the RL training. The demonstrations can be collected from asynchronous reanalysis of an expert solver, enabling RL to synergize with expert knowledge. Specifically, we conduct a decoupled training approach, employing two separate policy networks, RL and expert. During the Monte Carlo Tree Search (MCTS) process, action candidates from the expert policy foster a guided search mechanism, which is especially helpful in the early training stage. This framework alleviates the speed bottleneck typical of physics-based solvers in online decision-making, and also significantly enhances control performance and convergence speed of RL scheduling agents, as validated by substantial improvements in a 126-node real provincial test case.**

***Index Terms*—Real-time Scheduling, Reinforcement Learning from Demonstration, Predictive Control, Monte Carlo Tree Search**

I. Introduction

Reinforcement Learning (RL) has emerged as a promising alternative to computationally demanding optimization techniques for facilitating real-time scheduling and control [1], [2]. Despite its promise, the nascent RL solutions present challenges. Pure RL methods require careful design of reward functions [3], extensive interactions with the environment, and a long training time before convergence. Moreover, the performance gaps and lack of interpretability drive the studies about learning-assisted optimization [4], including unit commitment [5], [6] and optimal power flow [7]. However, since these methods require physics-based optimizers at each decision step, the decision time still depends on the quality of the warm-start solution and the problem scale. It is necessary to devise an elegant methodology combining physics-based optimizers and RL algorithms.

Reinforcement learning from demonstration (RLfD) aims at addressing the above problems, leveraging demonstration data to bolster online learning [8]. RLfD has broadened from vector-state tasks [9] to visual input [10], [11], action-free videos [12], and combinations with recently surged foundation models [13]. Many previous works have harnessed demonstration data by incorporating it into the replay buffer to accelerate training [14], [15]. However, offline demonstrations require scrupulous selection to avoid out-of-distribution issues. Some other works split training into two phases, imitation and RL fine-tuning [9], [16]. Nevertheless, the two-staged training cannot guarantee policy improvements after the imitation phase. The most related work in power systems is Deep Deterministic Policy Gradient from Demonstrations (DDPGfD) for service restoration [17]. DDPGfD uses behavior cloning to accelerate training and utilizes demonstration data to stabilize fine-tuning. The service restoration task shares a similar mixture of continuous actions and discrete actions as our power scheduling task. However, DDPGfD uses a two-staged training mechanism that might catch policy instability in the RL fine-tuning phase as previously mentioned. The model-free RL framework might also be constrained due to the lack of prospective planning capability.

In this regard, this paper develops a planning-capable RLfD solution for a complex power scheduling task, including optimal power flow (OPF) and unit commitment (UC). We are committed to employing advanced RL algorithms combined with RLfD technologies to approximate the performance of physics-based optimizers, while preserving the computation speed of learning-based methods. The upsurging renewable generation requires fast and precise decisions on not only the generators' power setpoints, but also the on-off statuses for fast start-stop generators. Consequently, the OPF and UC are formulated as a single unified Markov Decision Process (MDP). The designed RLfD operator observes the operational states (generator power, load injections, line flows, etc), and takes mixed actions (continuous power setpoints and discrete generator startup/shutdowns) to ensure economical and feasible power flow. In our 126-node test grid, a total of 54 generators are controllable, corresponding to 54-dimensional continuous control and 2^{54} possible combinations of generator status, which brings difficulties of high-dimensional control and combinatorial explosion.

This paper studies the following questions:

1) *Is it possible to achieve RLfD in power scheduling?*

[1]Institute for Interdisciplinary Information Science, Tsinghua University, Beijing 100084, email: liushaohuai42@gmail.com, gaoyangiiis@mail.tsinghua.edu.cn. [2]National Power Dispatching and Control Center, State Grid Corporation of China, Beijing 100031. [3]China Electric Power Research Institute, Beijing 100192. [4]Department of Electrical Engineering, Tsinghua University, Beijing 100084. *corresponding author.

This work was supported by State Grid Corporation of China Science and Technology Program "Research on Key Technologies for Static Security Risk Prevention and Control Based on Safe Reinforcement Learning" (No.5108-202355440A-3-2-ZN).

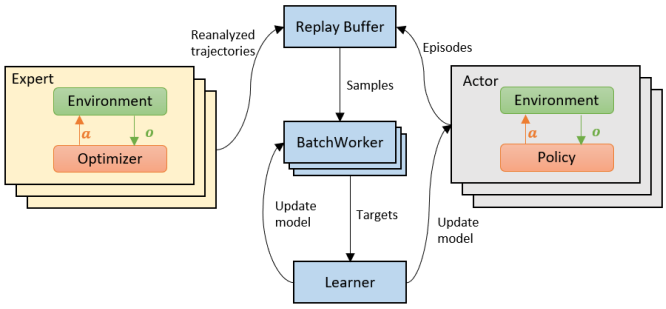

Fig. 1. Architecture of GridZero-Imitation (GZ-I).

2) *How to utilize demonstrations to accelerate training and stabilize fine-tuning in an end-to-end way?*

At a high level, this paper proposes an end-to-end framework that utilizes a mixture of imitation and RL fine-tuning. This work employs the physics-based optimizer to asynchronously reanalyze the scheduling trajectories. Hence, the fresh demonstrations continuously flow into the replay buffer. Simultaneously, the agent utilizes environmental rewards to further fine-tune the performance. An overview of our approach is described in Figure 1. Notably, the expert uses optimizers and not learnable.

The proposed RL agent is instantiated in the *GridSim* environment, which is similar to OpenAI Gym [18]. *GridSim* simulates power flows and provides interfaces for RL agents. The agent deployed on GridSim controls the generators' power setpoints and on-off status of fast startup units. The generation capability of renewable generators could vary with time. GZ-I agent is built on top of the previous GridZero [19], demonstrating the improved performance aided by RLfD.

Our contributions are formulated as follows:

- Different from the pure RL setting of GridZero [19], our method proposes an asynchronous framework enabling physics-based solvers to reanalyze the scheduling trajectories without speed limitations in online decisions. Fresh demonstrations are continuously generated, hence much less distribution shifts need to be considered.
- We employ guided search, as well as environmental rewards, which stabilize RL training and alleviate the affects of sub-optimal demonstrations due to accelerations for model linearizations and constraint relaxations.
- Different from GridZero [19], we make the GridSim environment more realistic by adding a preheating time for the startup process of thermal generators, which means thermal generators must wait for steps to output power after receiving the restart command.

Our work provides a natural combination of data-driven RL and traditional optimization methods, combining their respective strengths and avoiding weaknesses. This takes a step towards demonstration-assisted RL in power system applications. Compared to existing pure optimization-based methods, our method can accelerate over 100x times and achieve high renewable energy integration with only short-term forecasts (in 2 hours). Details are presented in Table I.

Our paper is organized as follows. Section II describes the preliminaries and how to formulate power scheduling as an MDP. Section III describes our efficient manner of incorporating demonstrations and fine-tuning. Section IV describes the experimental results. Section V is the discussion and conclusion.

II. Power Scheduling as a Markov Decision Process

In this section, we will first introduce the power scheduling problem definition. Subsequently, we formulate the power scheduling problem as a Markov Decision Process (MDP). Finally, we briefly introduce the preliminaries of GridZero [19].

A. Power Scheduling Problem

This work considers a power system where $\mathcal{N}, \mathcal{G}, \mathcal{L}$ indicate sets of nodes, generators, and lines. The active/reactive power outputs of generators at node $i \in \mathcal{N}$ are denoted as $p_{g,i}, q_{g,i}$. Non-operating generators at node i are simply defined as $p_{g,i} = q_{g,i} = 0$. The demand at node i is noted as $p_{d,i}, q_{d,i}$. This paper looks into a power grid operational setting where the power outputs and on-off status of generators can be adjusted for economical operations. Power output setpoints are constrained within time-varying bounds as $p^t_{g,i} \in [\underline{p^t_{g,i}}, \overline{p^t_{g,i}}]$.

Power scheduling consists of OPF and UC. For mathematical definitions, the OPF problem could be formulated as

$$
\begin{aligned}
\min_{p} \quad & \sum_{i \in \mathcal{B}} C(p_{g,i}) \\
s.t. \quad & p_{ij} = g_{ij}(v_i^2 - v_i v_j \cos\theta_{ij}) - b_{ij} v_i v_j \sin\theta_{ij} \\
& q_{ij} = b_{ij}(-v_i^2 + v_i v_j \cos\theta_{ij}) - g_{ij} v_i v_j \sin\theta_{ij} \\
& p^g_i - p^d_i = \textstyle\sum_j p_{ij}, \quad i \in \mathcal{N} \\
& q^g_i - q^d_i = \textstyle\sum_j q_{ij}, \quad i \in \mathcal{N} \\
& \underline{\theta_{ij}} \le \theta_{ij} = \theta_i - \theta_j \le \overline{\theta_{ij}}, \quad (i,j) \in \mathcal{L} \\
& p^2_{ij} + q^2_{ij} \le \overline{s_{ij}}, \quad (i,j) \in \mathcal{L} \\
& \underline{p_{g,i}} \le p_{g,i} \le \overline{p_{g,i}}, \quad i \in \mathcal{N} \\
& \underline{q_{g,i}} \le q_{g,i} \le \overline{q_{g,i}}, \quad i \in \mathcal{N} \\
& \underline{v_i} \le v_i \le \overline{v_i}, \quad i \in \mathcal{N}
\end{aligned}
\tag{1}
$$

where g_{ij}, b_{ij} indicate the conductance and susceptance of line (i,j). p_{ij}, q_{ij} represent the active and reactive power flowing on line (i,j). θ_{ij} represents the phase angle.

The UC problem could be defined as

$$
\begin{aligned}
\min_{u,p} \quad & \sum_{t=1}^{T} \sum_{g \in \mathcal{G}} c^s_i |u^t_g - u^{t-1}_g| + u^t_g C(p^t_g) \\
s.t. \quad & \textstyle\sum_{g \in \mathcal{G}} u^t_g p^t_g = L^t, \qquad t \in [0,T] \\
& \underline{p_g} u^t_g \le p^t_g \le \overline{p_g} u^t_g, \quad \forall g \in \mathcal{G}, t \in [0,T] \\
& u^t_g \in \{0,1\}
\end{aligned}
\tag{2}
$$

where L^t denotes the total load consumption at timestep t. u^t_g is the utility of generator g at timestep t, 1 as in-service and 0 for out-of-service. Only the total load-generation constraint is

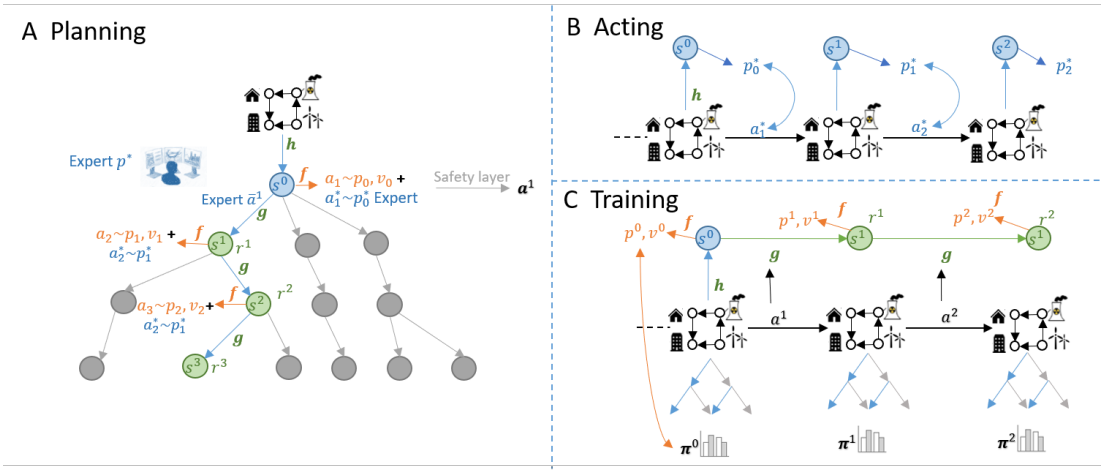

Fig. 2. Planning, Acting and Training of GridZero-Imitation.

demonstrated for simplified illustration, while all operational constraints and rules are considered in implementations of *GridSim* environment and ACOPF+UC baseline in Gurobi solver.

B. Reinforcement learning for power scheduling

Reinforcement learning has provided a powerful paradigm, by training a policy that maps the states to actions, to minimize the loss function defined in Equation 1 and 2. In the remainder of this section, we outline how the power scheduling problem defined in II-A can be formulated as an RL problem. To be specific, a Markov Decision Process (MDP) of $(\mathcal{S}, \mathcal{A}, \mathcal{T}, r, \gamma)$ represents the power scheduling model. $\mathcal{S}$ is the set of states (including generation and load consumptions, and line flows), $\mathcal{A}$ is the action space, $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ is the state transition function (reflecting system dynamics), $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, and γ is the discount factor.

State. The state consists of the following parts.

1) *Generators*. The agent observes the generator's information, including active/reactive power setpoints, adjustable ranges, as well as startable and closable indexes.
2) *Loads*. Load information includes active and reactive power.
3) *Lines*. Line information includes active/reactive transmission power and line currents.
4) *Nodes*. Nodal information consists of active/reactive power injections and voltages.
5) *Forecasts*. Forecasts include short-term predictions of load and renewable maximum power (corresponding to future 20 steps).

Action. For our RL agent, action $a = (a_p, a_o, a_c)$ consists of two parts:

1) *Dispatching* a_p. $a_p = \{\Delta p_{g,i}\}$. The agent can modify the generators' active power setpoints $p_{g,i}$ through $\Delta p_{g,i}$, $\forall i \in \mathcal{N}$. For thermal generators, $\Delta p_{g,i} \in \left[-c_{\text{ramp}} p_{g,i}^{\max}, c_{\text{ramp}} p_{g,i}^{\max}\right]$, where $c_{\text{ramp}} = 0.05$ in our experiments. Specifically, for closed generators, $\Delta p_{g,i} \in \{0\}$. For startable and closable generators, $\Delta p_{g,i} \in [0, p_{g,i}^{\min}]$ and $[-p_{g,i}^{\min}, c_{\text{ramp}} p_{g,i}^{\max}]$. For renewable generators, $\Delta p_{g,i} \in [-p_{g,i}, \overline{p_{g,i}} - p_{g,i}]$.
2) *Startup/shutdown* a_o, a_c. The startup/shutdown actions are set to be one-hot vectors, which means no more than one generator can be started or shut down within one decision step. The dimensions of a_o, a_c are $|\mathcal{G}|+1$, where the last dimension indicates doing nothing (no shutdown or startup). Renewable generators are assumed to always be in service. The illegal startup/shutdown action will be masked out by the startable/closable indexes provided in states.

Reward. Operational constraints, such as nodal voltage, line current, and reactive power limits, are all ranged constraints, limiting the variables within feasible conditions. These constraints are presented in the form of reward functions. The total reward is the sum of the reward parts defined by operational objectives and constraints. Further design details are provided in Appendix V-A.

$$r = \underbrace{\lambda_1 r_{\text{renewable}} + \lambda_2 r_{\text{cost}}}_{\text{objectives}} + \underbrace{\lambda_3 r_{\text{overflow}} + \lambda_4 r_{\text{reactive}} + \lambda_5 r_{\text{voltage}}}_{\text{constraints}} \tag{3}$$

State Transition. Despite power flow simulations, we simulate realistic operational rules within state transitions. The environment has different counters to manage these operational rules. When these events are triggered, the counter will be set to a value and start counting down. Once the counter reduces to 0, the action space will change or the reconnection will be automatically conducted. To be specific, if a generator is just shut down, a counter will be set to N (N varies according to the capacity). Within the following N steps, the action space of this closed generator is set to $\{0\}$, hence preventing restarting this generator. Transmission lines would be disconnected if exceeding overflow limits. All disconnected lines would be reconnected automatically after maintenance steps.

C. Model-based RL for power scheduling

GridZero is the first attempt to apply DNNs and MCTS in power scheduling tasks [19]. Similar to GridZero, GZ-

I has two functions, prediction $f(s) \rightarrow p, v$, and dynamics $g(s,a) \rightarrow s', r$. The policy p employs a Gaussian distribution $p_{\mathcal{N}}$ and two categorial distributions p_o, p_c to sample continuous adjustments of power setpoints a_p and discrete actions on thermal generator status a_o, a_c, due to high-dimensional continuous action space and the huge number of unit combinations. Sampled actions are set as action candidates of each node in MCTS as Samped MuZero [20].

In MCTS, the agent selects actions via the UCB formula, which is defined as

$$a = \arg\max_{a \sim p} \overbrace{Q(s,a) + \underbrace{\hat{\beta}(s,a)}_{\text{uniform prior}} \frac{\sqrt{\sum_b N(s,b)}}{1+N(s,a)}}^{\text{UCB score}} \tag{4}$$

where $Q(s,a)$ is the action-value of transition (s,a). $\hat{\beta}$ is a uniform distribution since the preferences of policies are reflected in distributions in the action sampling setting [20]. $N(s,a)$ indicates the number of taking action a at state s, and $N(s,b)$ refers to visit counts of sibling nodes.

As demonstrated in Figure 2(A) and (B), the agent dives to a leaf node via the path with the highest UCB scores, then expands new leaves and updates the values along the search path. This process is repeated to grow a search tree as shown in Figure 2(B).

The after-search root visit count distribution π and sampled actions a are set as the training target of policy function p. The reward prediction r is supervised by environmental rewards u. The value prediction v is supervised by the N-step bootstrapped target z. The dynamics function is supervised by self-consistency as EfficientZero [21]. GridZero loss can be defined as follows

$$\begin{aligned} l_{base} = \sum_{k=0}^{K} [& \underbrace{l(u_k, r_k)}_{\text{reward}} + \underbrace{l(z_k, v_k)}_{\text{value}} + \underbrace{l(\pi_k, p_k, a_k)}_{\text{policy}} \\ & + \underbrace{l(s_k, \hat{s}_k)}_{\text{dynamics}} - \underbrace{H(p_k)}_{\text{entropy}}] \end{aligned} \tag{5}$$

where $z = \sum_{i=0}^{N=5} \gamma^i u_i + \gamma^5 \hat{v}(s_5)$ refers to the target value. $H(p_k)$ is the entropy of the policy function, used for encouraging exploration. The policy loss is defined as

$$\begin{aligned} l(\pi, p, a) = & -\pi^\top \cdot \log p_{\mathcal{N}}(a_p) \\ & - (\pi \cdot a_o)^\top \cdot \text{LogSoftmax}(p_o) \\ & - (\pi \cdot a_c)^\top \cdot \text{LogSoftmax}(p_c) \end{aligned} \tag{6}$$

III. Proposed efficient imitation and stable fine-tuning

This section first discusses how to incorporate the traditional optimizer to asynchronously reanalyze the post-scheduled trajectories. We then establish an end-to-end framework that combines behavior cloning (imitation) with online fine-tuning. We introduce the mixed loss function design, and formulate a guided tree search mechanism.

A. Expert experience collection through asynchronous reanalysis

Different from holding a fixed demonstration buffer as conventional RLfD algorithms, this work creates a continuous expert data flow by employing the optimizer to produce expert actions for the RL-collected trajectories. Our method allows cold-start, which means starting training without initial expert demonstrations. We construct the optimizer through the combination of OPF solvers and UC solvers. For the optimal power flow problem, we choose the common interior point solver (IPS), and consider AC (considering voltage and reactive power constraints) and DC (linearized model for acceleration) settings [22]. In context of future applications in large networks, we investigate the influences on the performance of taking optimal (AC-OPF) and suboptimal (DC-OPF) demonstrations in Section IV-C.

The UC problem is solved as the common mixed integer programming (MIP) modeling. However, the MIP problem is computationally intensive, making it expensive for asynchronous demonstration generation. This work takes an alternative heuristic method modified from PyPower *duopf* function [23], as demonstrated in Algorithm 1.

Algorithm 1 Heuristic Unit Commitment (HUC, Modified from *duopf*)

Require:
Total load L^{t+i}, maximum total renewable generation $\overline{p_{g,r}^{t+i}}, i \in [0,10]$, $a_p \in \mathbb{R}^{|\mathcal{G}|}$, $a_o, a_c \in \{0,1\}^{|\mathcal{G}|+1}$

while $p_g^t > L^{t+1}$ **do**
  Find the unit with the lowest shutdown fee i
  $a_c[i] = 1$,
end while
while $\overline{p_g^t} < L^{t+10} - (\overline{p_{g,r}^{t+10}} - \overline{p_{g,r}^t})$ **do**
  Find the unit with the lowest startup fee i
  $a_o[i] = 1$, break
end while
for g in ready-to-start generators **do**
  if OPF($\mathcal{G} \cup g$) < OPF($\mathcal{G}$) **then**
    $a_o[g] = 1$, $\mathcal{G} \leftarrow \mathcal{G} \cup g$, break
  end if
end for
for g in closable generators **do**
  if OPF($\mathcal{G} \setminus g$) < OPF($\mathcal{G}$) **then**
    $a_c[g] = 1$, $\mathcal{G} \leftarrow \mathcal{G} \setminus g$, break
  end if
end for
$a_p = \text{OPF}(\mathcal{G})$
return a_p, a_o, a_c;

The Heuristic Unit Commitment (HUC) only considers the balancing of total load and total generation, without considering network constraints. Hence its decisions are much faster, but the obtained solutions are suboptimal. The core ideas are as follows. If the maximum power is lower than the future 10-step load consumption, one ready-to-start generator

Algorithm 2 GridZero-Imitation (GZ-I)

Require:
Training Steps N, traditional optimizer HUC, initial parameter θ_0

Start self-play workers to interact with env and collect trajectories $\{(s_t, a_t, r_t, s_{t+1})\}$.
Start expert workers to relabel expert actions (s_t, a_t^*) using HUC optimizer.
Start batch workers to sample transitions from the replay buffer and make learning targets.
for $i = 1, \cdots, N$ **do**
  $\theta_i \leftarrow \theta_{i-1}$
  Get training batches from batch workers and calculate loss l via Equation 8.
  $\theta_i \leftarrow \theta_i + \nabla l(\theta_i)$
  if reach update interval **then**
    update θ_i to self-play workers and batch workers.
  end if
end for
return θ_N;

is forced to be started. For the minimum power exceeding future load consumption, HUC takes similar actions that close one generator. In case of total load remains within the range of minimum power to maximum power, HUC executes a heuristic search within all closable and ready-to-start generators to find a minimum operating cost. This is implemented by enumeration that turning on or off the generators one by one and performing OPF calculations. Although HUC results are suboptimal to MIP results, our approach can further fine-tune towards optimal solutions with the guidance of environmental rewards.

The constructed optimizer continuously samples post trajectories, reanalyzes each state transition with expert actions $s_t \rightarrow a_t^*$, and sends reanalyzed trajectories to the replay buffer.

B. Efficient imitation and stable RL fine-tuning combined with Monte Carlo Tree Search

How to train policy with the aid of demonstrations? Given the reanalyzed expert demonstrations, a natural idea is to train the RL agent with the aid of expert policies. Essentially, the demonstration data can serve as a regulator so that the RL policy will not deviate far from the expert policy. Mathematically, this regulator can be defined as a regularization loss:

$$l = l_{base} + l_{bc} \tag{7}$$

where l_{base} is the GridZero's loss. l_{bc} indicates the behavior cloning regularization loss. Different from most previous RLfD works that share a single policy head in behavior cloning and RL fine-tuning, we design a separate expert policy head $\bar{p}$ for decoupling gradients of different objectives. As Figure 2 demonstrates, the expert policy is supervised by the reanalyzed expert action a^*. The l_{bc} loss is defined as maximizing the likelihood of demonstration actions and the expert policy:

$$l_{bc} = \sum_{k=0}^{K} l_{bc}(\bar{p}_k, a_k^*) \tag{8}$$

where a^* is the concatenation of power setpoints adjustments a_p^*, startup one-hot vector a_o^*, and shutdown one-hot vector a_c^*, which is the same as the RL action a. In the same way as the RL policy p, the expert policy $\bar{p}$ consists of a Gaussian policy $\bar{p}_{\mathcal{N}}$, a startup categorial policy $\bar{p}_o$, and a shutdown categorial policy $\bar{p}_c$. Hence the l_{bc} can be further formulated in mathematics as:

$$\begin{aligned} l_{bc}(\bar{p}, a^*) = &- \log \bar{p}_{\mathcal{N}}(a_p^*) \\ &- a_o^{*\top} \cdot \text{LogSoftmax}(\bar{p}_o) \\ &- a_c^{*\top} \cdot \text{LogSoftmax}(\bar{p}_c) \end{aligned} \tag{9}$$

Overall, the expert policy is trained as a supervised learning task. In the following, we demonstrate how the expert policy guides the RL policy.

How to use the expert policy to guide RL policy and stablize fine-tuning? As mentioned in Section II-C, training targets of the RL policy come from the root visit count distributions of the MCTS process. Therefore, the expert policy is supposed to be incorporated into the search process, and have impacts on the final visit count distribution target π.

Specifically, action candidates at each node are sampled from both the RL policy and the expert policy, as demonstrated in Figure 2. The proportion is set as $4:1$ in our experiments. In this way, the UCB formula will choose expert actions with higher probabilities if the RL policy deviates significantly from the expert policy. It thus changes the final RL policy target because the expert actions have more visit counts. This forms a guided search mechanism that allows expert experience to be effectively transferred into RL policies. Compared to the commonly used 'single network-multi loss' and teacher-student network momentum updating methods, our method is an explicit transferring with better interpretability.

Meanwhile, the demonstrations used are not theoretically optimal due to computation speed limitations. Therefore, the agent still needs the environmental reward to provide further training signals for RL fine-tuning. The search process will lean towards RL actions if the RL policy is better than the expert policy. At this point, the agent will trade off between RL fine-tuning and BC regularization.

IV. Evaluation

We end up this paper with experiments demonstrating the effectiveness of our proposed approach. The first is our evaluation setup and the results of our efficient learning framework compared to baselines. We then investigate the influences of different quality of demonstrations. Finally, this work briefly demonstrates the scenarios containing topology changes.

A. Environment and Evaluation

Experimental Setup. A real provincial network is deployed to evaluate our results. The grid has 126 buses, 185 power lines, and 54 generators (17 renewables), resulting in 54-dimensional continuous control and 2^{54} possible unit commitments. Our test grid uses the same parameters as the real grid, it is reasonable to expect our capability to be maintained in real-world tasks.

Different from some previous works using the same operational sections as the training and test datasets, this work uses two-year data as separate datasets, which specify the load and renewable generation data every 5 minutes. Each scenario is a one-day power scheduling task, including 288 decision steps. During training, the environment can randomly start from any section of the training set. It starts from the specified sections while testing.

The thermal generators have ramping ratios of 0.05, while renewable generators do not. The start-close minimal interval is set as 40 decision steps. The ready-to-start generator can be restarted in 10 decision steps after receiving the restart command. Transmission lines would be disconnected if entering soft overflow for 6 steps, and get immediately disconnected if entering hard overflow. The soft overflow and hard overflow limits are set as 1.0 and 1.25 times of maximum line loading current. The disconnected line will be recovered automatically after 16 maintenance steps.

Our Method and Baselines. To study the effectiveness of our method, the ACOPF+UC is employed as the baseline of physics-based optimization, implemented by PyPower [23] and Gurobi [24] respectively. The UC model considers network constraints in problem formulation. We also set the simple behavior cloning (BC) as one of the baseline. The most related DDPGfD algorithm is set as an RLfD baseline which shares the same setting of mixing imitation and RL fine-tuning but with a model-free RL backbone [17]. Finally, we use GridZero (pure RL) to demonstrate our method can further stabilize the RL fine-tuning.

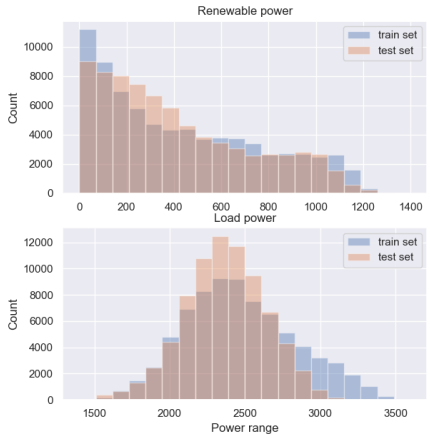

Fig. 3. Histograms of renewable and load power distributions.

B. Efficient imitation and stable RL finetuning

We present results compared with ground-truth ACOPF+UC, pure behavior cloning (BC), and an RLfD baseline DDPGfD. We also compare GridZero to demonstrate that our mixed training and guided search can stabilize RL fine-tuning.

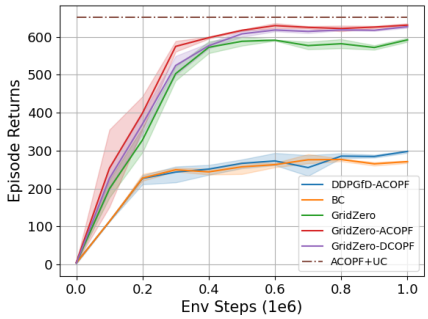

Fig. 4. Training curves of DDPGfD, Behavior Cloning (BC), *GridZero*-ACOPF and *GridZero*-DCOPF. The proportion of reanalyzed transitions and self-play transitions is set to 1 : 10.

Compared with ACOPF+UC and BC. As demonstrated in Figure 4, the ground-truth expert using mixed integer programming and optimal power flow solver can achieve an average score of 649.4 in all test scenarios. Our proposed RLfD framework *GridZero-imitation* can achieve an average score of 628.3 with HUC and ACOPF demonstrations. However, the BC method only has a much lower average score. This is mainly due to the distribution deviation between the training set and the test set as shown in Figure 3. Behavior Cloning (BC) is sensitive to such distribution shifts which cause policy collapses while testing.

TABLE I
TEST STATISTICS OF ACOPF+UC AND *GridZero*-ACOPF (10 SEEDS AVERAGE).

	BC	DDPGfD	ACOPF+UC	GZ-I
T_{episode}(s)	19.2	20.9	8557.3	43.2
Episodic Rewards	278.3	299.7	649.4	628.3
$\lvert V_{\text{bus}}\rvert$ violation(%)	0.9	0.8	0.3	0.5
Q violation(%)	14.4	16.1	6.2	6.5
P_{balanced} violation(%)	0.0	0.0	0.0	0.0
Line soft overflow(%)	11.2	10.8	6.5	6.9
Line hard overflow(%)	0.0	0.0	0.0	0.0
Operating cost(K $)	50356	48785	38565	40932
Renewable consumption(%)	61	64	97	93
N-1 tolerance rate(%)	89	87	95	96

Compared to DDPGfD. We take DDPGfD as the RLfD baseline. Our proposed method outperforms DDPGfD with the same environmental steps and demonstrations. This indicates the benefits brought by model-based planning. DDPGfD employs a single actor-network and estimates policy gradients both from the estimated Q-value with high variances and the BC loss. Conversely, we employ four decoupled supervised trainings of reward, value, RL policy, and expert policy functions. The advantage of decoupled training is that each network has only one supervised signal and is not affected

by other loss functions. We incorporate the four well-trained networks within an explicit planning process and produce a distilled search policy. Such decoupled supervised learning and explicit unified search are the essential characteristics that distinguish our method from others. This also makes our method outperform DDPGfD as shown in Figure 4.

C. Different qualities of demonstrations

ACOPF vs. DCOPF. We investigate the influences of demonstrations with different qualities. In large grids, the solving process of ACOPF demonstrations could be extremely time-consuming or even without feasible solutions. Model linearization is an acceptable trade-off between performance and computation speed. In this case, we study if downgraded demonstrations will affect the fine-tuning performance. As demonstrated in Figure 4, the differences in qualities of demonstrations didn't affect the final fine-tuning performance despite slightly slower convergence at the beginning. This indicates the robustness of our algorithm to different demonstrations, which is critical for applicability to larger grids.

D. Topology changes

N-1. The potential component failures that threaten the safe operations of power grids, like line outages, can change the operational topology. Such failures change the power flow and invalidate the pre-calculated scheduling solutions. The previous works propose that topology changes would affect general learning-based methods, leading to performance drops [25]. We demonstrate such a performance drop can be avoided by adding topology information (including line current magnitudes and directions, etc.) into observations and randomly attacking the agent by disconnecting one line with a certain probability while collecting data. As shown in Figure 5, an line outage lasted for 16 decision steps, and the agent successfully ran to the reconnection. However, some outages caused cascading failures. As shown in Table I, there is still a 4% probability that the agent failed to deal with N-1. In future work, we will focus on reducing the probability of such occurrences.

V. Conclusion

In this paper, we study how to combine physics-based optimization methods with advanced data-driven RL. We aim to improve the performance of RL policy and alleviate the affects of computing speed bottlenecks of optimization on decision-making. By using decoupled training and explicit forward search, we create a framework that exploits the respective advantages of optimization and RL. Our method has the ability to be extended to more tasks, especially in scenarios where optimization is slow but requiring fast decision-making.

We further provide a realistic test case, which suggests that this effective Reinforcement Learning from Demonstrations technique has the potential to be leveraged in real-world applications. We also demonstrate that our agent can withstand the topology changes to a certain extent. However, deliberate attacks against weak links and their defenses need further discussions in future adversarial training works.

References

[1] Y. Zhou, W.-J. Lee, R. Diao, and D. Shi, "Deep reinforcement learning based real-time ac optimal power flow considering uncertainties," *Journal of Modern Power Systems and Clean Energy*, 2021.

[2] J. H. Woo, L. Wu, J.-B. Park, and J. H. Roh, "Real-time optimal power flow using twin delayed deep deterministic policy gradient algorithm," *IEEE Access*, vol. 8, pp. 213611–213618, 2020.

[3] Y. Zhou, B. Zhang, C. Xu, T. Lan, R. Diao, D. Shi, Z. Wang, and W.-J. Lee, "A data-driven method for fast ac optimal power flow solutions via deep reinforcement learning," *Journal of Modern Power Systems and Clean Energy*, vol. 8, no. 6, pp. 1128–1139, 2020.

[4] G. Ruan, H. Zhong, G. Zhang, Y. He, X. Wang, and T. Pu, "Review of learning-assisted power system optimization," *CSEE Journal of Power and Energy Systems*, vol. 7, no. 2, pp. 221–231, 2020.

[5] Y. Yang and L. Wu, "Machine learning approaches to the unit commitment problem: Current trends, emerging challenges, and new strategies," *The Electricity Journal*, vol. 34, no. 1, p. 106889, 2021.

[6] Á. S. Xavier, F. Qiu, and S. Ahmed, "Learning to solve large-scale security-constrained unit commitment problems," *INFORMS Journal on Computing*, vol. 33, no. 2, pp. 739–756, 2021.

[7] Y. Cao, H. Zhao, G. Liang, J. Zhao, H. Liao, and C. Yang, "Fast and explainable warm-start point learning for ac optimal power flow using decision tree," *International Journal of Electrical Power & Energy Systems*, vol. 153, p. 109369, 2023.

[8] S. Schaal, "Learning from demonstration," *Advances in neural information processing systems*, vol. 9, 1996.

[9] T. Hester, M. Vecerik, O. Pietquin, M. Lanctot, T. Schaul, B. Piot, D. Horgan, J. Quan, A. Sendonaris, I. Osband, *et al.*, "Deep q-learning from demonstrations," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 32, 2018.

[10] K. Pertsch, Y. Lee, Y. Wu, and J. J. Lim, "Guided reinforcement learning with learned skills," *arXiv preprint arXiv:2107.10253*, 2021.

[11] D. Rengarajan, G. Vaidya, A. Sarvesh, D. Kalathil, and S. Shakkottai, "Reinforcement learning with sparse rewards using guidance from offline demonstration," *arXiv preprint arXiv:2202.04628*, 2022.

[12] W. Ye, Y. Zhang, P. Abbeel, and Y. Gao, "Become a proficient player with limited data through watching pure videos," in *The Eleventh International Conference on Learning Representations*, 2022.

[13] S. Yang, O. Nachum, Y. Du, J. Wei, P. Abbeel, and D. Schuurmans, "Foundation models for decision making: Problems, methods, and opportunities," *arXiv preprint arXiv:2303.04129*, 2023.

[14] M. Vecerik, T. Hester, J. Scholz, F. Wang, O. Pietquin, B. Piot, N. Heess, T. Rothörl, T. Lampe, and M. Riedmiller, "Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards," *arXiv preprint arXiv:1707.08817*, 2017.

[15] A. Nair, B. McGrew, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Overcoming exploration in reinforcement learning with demonstrations," in *2018 IEEE international conference on robotics and automation (ICRA)*, pp. 6292–6299, IEEE, 2018.

[16] G. Krishnamoorthy, A. Dubey, and A. H. Gebremedhin, "Reinforcement learning for battery energy storage dispatch augmented with model-based optimizer," in *2021 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)*, pp. 289–294, IEEE, 2021.

[17] Y. Du and D. Wu, "Deep reinforcement learning from demonstrations to assist service restoration in islanded microgrids," *IEEE Transactions on Sustainable Energy*, vol. 13, no. 2, pp. 1062–1072, 2022.

[18] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," *arXiv preprint arXiv:1606.01540*, 2016.

[19] S. Liu, J. Liu, W. Ye, N. Yang, G. Zhang, H. Zhong, C. Kang, Q. Jiang, X. Song, F. Di, *et al.*, "Real-time scheduling of renewable power systems through planning-based reinforcement learning," *arXiv preprint arXiv:2303.05205*, 2023.

[20] T. Hubert, J. Schrittwieser, I. Antonoglou, M. Barekatain, S. Schmitt, and D. Silver, "Learning and planning in complex action spaces," in *International Conference on Machine Learning*, pp. 4476–4486, PMLR, 2021.

[21] W. Ye, S. Liu, T. Kurutach, P. Abbeel, and Y. Gao, "Mastering atari games with limited data," *Advances in Neural Information Processing Systems*, vol. 34, pp. 25476–25488, 2021.

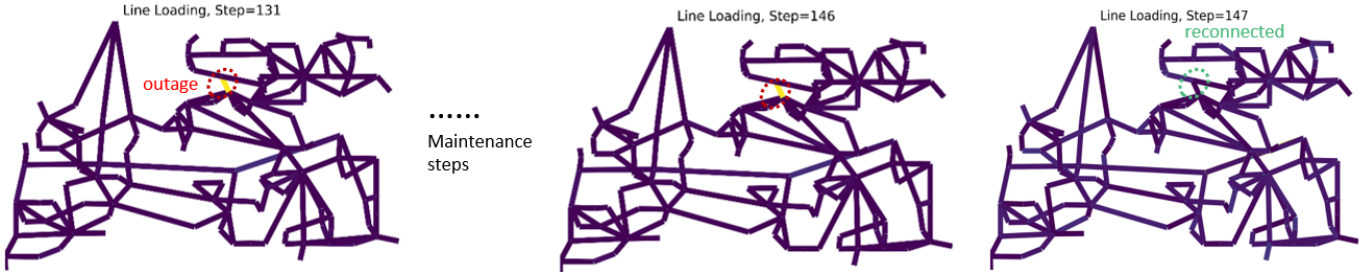

Fig. 5. N-1 outage occurrence and recovery process.

[22] R. A. Jabr, A. H. Coonick, and B. J. Cory, "A primal-dual interior point method for optimal power flow dispatching," *IEEE Transactions on Power Systems*, vol. 17, no. 3, pp. 654–662, 2002.
[23] R. Lincoln, "Pypower: Port of matpower in python," *URL: https://github.com/rwl/PYPOWER, last accessed on*, vol. 25, no. 12, p. 2019, 2019.
[24] L. Gurobi Optimization, "Gurobi optimizer reference manual," 2021.
[25] M. Zhou, M. Chen, and S. H. Low, "Deepopf-ft: One deep neural network for multiple ac-opf problems with flexible topology," *IEEE Transactions on Power Systems*, vol. 38, no. 1, pp. 964–967, 2022.

APPENDIX

A. Reward design details

The weight coefficients are $\lambda_1 = 2, \lambda_2 = 4, \lambda_3 = 1, \lambda_4 = 2, \lambda_5 = 1$ in practice. The following are more details about each component.

Renewable consumption. More renewable energy consumption corresponds to a higher reward.

$$r_{\text{renewable}} = \frac{\sum_i \mathrm{p}_i}{\sum_i \mathrm{p}_i^{\max}}, \quad i \in \text{renewable units}$$

where p_i represents the power output of generator i, $\mathrm{p}_i^{\max}$ indicates maximal power generation capability of renewable generator i. The closer p_i and $\mathrm{p}_i^{\max}$ are, the higher the renewable consumption reward.

Operating cost. Operating costs of thermal generators are quadratic functions of output power, and additional costs are incurred for the startup/shutdowns. Operating costs of renewable sources are considered to be negligible.

$$r_{\text{cost}} = -\frac{\sum_i c_{i,2}\mathrm{p}_i^2 + c_{i,1}\mathrm{p}_i + c_{i,0} + \mathcal{I}(\mathrm{s}_i, \mathrm{s}_i^-)c_{\text{on-off,i}}}{Z}$$

where $c_{i,2}, c_{i,1}$ and $c_{i,0}$ are the second order, first order, and constant coefficients of the operation cost of generator i, respectively. The coefficients of renewable units are much lower than that of thermal units. p_i represents the power output of generator i. s_i represents the on-off status of generator i, and the s_i^- is the status 1-step advance. $c_{\text{on-off,i}}$ is the startup and shutdown costs of generator i. $\mathcal{I}(\mathrm{s}_i, \mathrm{s}_i^-)$ is an indicative function that turns to be 1 if $\mathrm{s}_i \neq \mathrm{s}_i^-$, otherwise 0. Z is the normalization factor set as 10^5 in experiments.

Line overflow. The current load rate is equal to the ratio of the transmission current to the maximum transmission current. Maintaining a reasonable load level is necessary to prevent line congestion and outages.

$$r_{\text{overflow}} = 1 - \frac{\sum_i \min(\rho_i, 1)}{n_{\text{line}}}$$

where ρ_i indicates the current load rate of line i. n_{line} represents the line number.

Reactive power. The reactive power output capacities of the generators are constrained. Exceeding the limits can significantly increase operational costs.

$$r_{\text{reactive}} = \exp\left(-\sum_i \left[\frac{\max(\mathrm{q}_i - \overline{\mathrm{q}_i}, 0)}{\overline{\mathrm{q}_i} - \underline{\mathrm{q}_i}} + \frac{\max(\underline{\mathrm{q}_i} - \mathrm{q}_i, 0)}{\overline{\mathrm{q}_i} - \underline{\mathrm{q}_i}}\right]\right) - 1$$

where q_i is the reactive power of generator i, and $\overline{\mathrm{q}_i}, \underline{\mathrm{q}_i}$ are the upper bound and the lower bound of generator i. There would be a penalty if any generator violates its reactive power constraint.

Bus voltage. To regulate the node voltage magnitudes within specified ranges, we design the bus voltage reward similar to the reactive power reward.

$$r_{\text{voltage}} = \exp\left(-\sum_i \left[\frac{\max(\mathrm{v}_i - \overline{\mathrm{v}_i}, 0)}{\overline{\mathrm{v}_i} - \underline{\mathrm{v}_i}} + \frac{\max(\underline{\mathrm{v}_i} - \mathrm{v}_i, 0)}{\overline{\mathrm{v}_i} - \underline{\mathrm{v}_i}}\right]\right) - 1$$

where v_i is the voltage magnitude of bus i, and $\overline{\mathrm{v}_i}, \underline{\mathrm{v}_i}$ are the upper bound and the lower bound of voltage magnitude of bus i. There would be a penalty if any bus violates its voltage magnitude constraint.

B. Hyper-parameters

TABLE II
HYPER-PARAMETERS FOR GRIDZERO-IMITATION

Parameter	Setting
Max frames per episode	288
Discount factor	0.997
Minibatch size	256
Optimizer	SGD
Optimizer: learning rate	0.01
Optimizer: momentum	0.9
Optimizer: weight decay (c)	1e-4
Max gradient norm	10
Priority exponent (α)	0.6
Priority correction (β)	0.4
Training steps	100k
Self-play network updating inerval	100
Target network interval	1000
Unroll steps (l_{unroll})	5
TD steps (k)	5
Number of simulations in MCTS (N_{sim})	50