Efficient Constraint Learning for Data-Driven Active Distribution Network Operation

Ge Chen, Graduate Student Member, IEEE, Hongcai Zhang, Senior Member, IEEE, and Yonghua Song, Fellow, IEEE

Abstract—Scheduling flexible sources to promote the integration of renewable generation is one fundamental problem for operating active distribution networks (ADNs). However, existing works are usually based on power flow models, which require network parameters (e.g., topology and line impedance) that may be unavailable in practice. To address this issue, we propose an efficient constraint learning method to operate ADNs. This method first trains multilayer perceptrons (MLPs) based on historical data to learn the mappings from decisions to constraint violations and power loss. Then, power flow constraints can be replicated by these MLPs without network parameters. We further prove that MLPs learn constraints by formulating a union of disjoint polytopes to approximate the corresponding feasible region. Thus, the proposed method can be interpreted as a piecewise linearization method, which also explains its desirable ability to replicate complex constraints. Finally, a novel pruning method is developed to remove the useless binary variable solutions in advance, which can enhance the solution’s reliability and reduce the computational complexity. Numerical experiments based on the IEEE 33- and 123-bus test systems validate that the proposed method can achieve desirable optimality, feasibility, and computational efficiency simultaneously.

Index Terms—Active distribution networks, deep learning, renewable generation, optimal power flow, flexible sources.

I. INTRODUCTION

MOTIVATED by the carbon neutrality, increasing distributed renewable generation (DRG) has been integrated into distribution networks [1]. The high DRG penetration promotes the transformation of passive distribution networks into active distribution networks (ADNs), which unlocks the potential flexibility from the distribution level [2].

For ADNs, one fundamental problem is how to coordinate their flexible sources, such as energy storage systems [3], electric vehicles [4], or heating, ventilation, and air conditioning (HVAC) systems [5], to reduce the curtailment of DRG. Traditionally, this coordination is guided by power flow models.

For example, reference [6] employed the second-order cone programming (SOCP) relaxation of the DistFlow model to coordinate heating systems with DRG. Based on this SOCP relaxation, reference [7] designed a distributed algorithm for managing the demand responses of HVAC systems in ADNs. Reference [8] also used this SOCP relaxation to design the charging/discharging scheduling strategy for electric vehicles to promote DRG integration in ADNs. Reference [9] combined a linearized power flow model with distributed optimization schema to schedule flexible sources in ADNs. In general, these model-based works can effectively operate ADNs. However, they usually require power network parameters (e.g., network topology and line impedance) that are often unavailable at the distribution level due to unaware topology changes or inaccurate data maintenance [10]. Moreover, some approximations (e.g., SOCP relaxations used in [6], [7], [8]) are usually introduced to convexify the non-convex power flow equations. However, these approximations may introduce considerable errors and even infeasible solutions [11].

Due to the popularization of smart meters, collecting operational data of ADNs (e.g., nodal power injections, bus voltages, and power flows) has become easier and cheaper [12]. Since this data contains the knowledge of network modeling, learning-based methods may be an alternative choice for operating ADNs [13]. Generally speaking, these learning-based methods can be divided into three categories.

1) Optimize-then-learn methods: Methods in this category first solve multiple instances of optimization problems to construct training sets, where the input features are operating conditions (e.g., power demands and available DRG) and the output labels are optimal solutions (e.g., schedules of flexible sources), respectively. Then, supervised learning models are trained to approximate the mapping from a specific condition to the optimal decision. For example, reference [14] trained multilayer perceptrons (MLPs) to predict the optimal generation based on demand conditions. References [15], [16] combined this method with the Lagrangian dual approach to improve its feasibility. Reference [17] replaced MLPs with recurrent neural networks to predict the optimal schedule of flexible sources. Since the above methods replace the solving process of optimization problems with neural networks’ inferences, they can achieve excellent computational efficiency.

2) Reinforcement Learning: Reinforcement learning (RL) trains agents on how to act to maximize the expected cumulative reward (e.g., the opposite of total operation costs) in the future.
Reference [18] used RL to realize the voltage control of distribution networks. Reference [19] leveraged the multi-agent RL to operate distribution networks, which can reduce communication costs required by network training. Reference [20] proposed a graph-based RL method to coordinate flexible sources for the restoration of distributed systems. Since RL can directly infer the best action based on current state information, it shows great potential in realizing the real-time operation of ADNs.

3) Constraint Learning: Constraint learning trains neural networks to learn and replicate complex constraints, e.g., non-convex power flow equations. For example, in references [21], [22], binary classifiers were trained as surrogate models of power flow models to judge the feasibility of a given decision. After training, the classifiers were equivalently reformulated as mixed-integer linear constraints that off-the-shelf solvers can easily handle. Instead of using binary classifiers, our previous work employed a regression multi-layer perceptron (MLP) to predict the maximum constraint violation so that the feasibility of solutions can be improved [23]. By replacing the MLP with a quantile regression network, we further extended constraint learning to chance-constrained scheduling problems for distribution networks [24]. This regression neural network-based constraint learning has also been applied to many different problems, such as power-flow-constrained unit commitment [25], oil production optimization [26], and extractive distillation unit optimization [27]. It only requires operational data instead of optimal solutions as training data, so it can bypass the requirement of network parameters. Moreover, desirable optimality can also be achieved since the global optima of the mixed-integer replication can be found by the Branch-and-Bound algorithm (B&B).

Although the above papers have confirmed the effectiveness of the learning-based methods, they are still facing some challenges. Specifically, the optimize-then-learn methods, including [14], [15], [16], [17], need optimal solutions as training labels. Thus, network parameters are still indispensable because they need to build power flow models to generate training labels. For RL-based methods, such as [18], [19], [20], their agents need to continuously interact with actual environments (e.g., distribution networks) to update their policies. This “trial and error” scheme can be risky and is usually unacceptable in practice. The constraint learning methods used in [21], [22], [23], [24] do not require network parameters or interactions with real systems. However, its interpretability is poor because neural networks are usually regarded as “black boxes.” Though some prior works, such as [28], have proved that a neural network can be interpreted as a piecewise linear function, most of them only focused on modeling tasks and did not consider the potential of neural networks for replicating constraints. In fact, to the best of our knowledge, there is no theoretical interpretation to explain how constraint learning replicates complex constraints. Meanwhile, the computational efficiency of constraint learning may be undesirable. As aforementioned, constraint learning reformulates the trained neural networks as linear constraints with binary variables to replicate constraints. References [21], [22], [23], [24] pointed out that the number of involved binary variables equals the total neuron number of neural networks.

When the original constraint is complex enough and needs a neural network with many neurons to replicate, B&B needs to compare numerous binary variable solutions to find the best one, leading to a huge computational burden. Moreover, reference [29] pointed out that constraint learning may extrapolate significantly from the training data, leading to a solution in an area where neural networks have not learned. In this case, the reliability of solutions can not be ensured.

On the basis of our previous works [23], [24], we propose an interpretable and more efficient constraint learning method to overcome the challenges above. The proposed method trains two MLPs to predict constraint violations and power loss. Then, power flow constraints can be replicated with no need for network parameters. Compared to our previous works [23], [24] and other existing methods, the specific contributions of this paper are twofold:

1) We prove that constraint learning can be interpreted as a piecewise linearization (PWL) technique. We first show that the trained neural network is a piecewise linear approximator that partitions the domain of decision variables into disjoint linear pieces. Each piece corresponds to one binary variable solution of constraint learning. In every piece, the neural network’s output is linearly dependent on decision variables. To the best of our knowledge, this is the first time that a theoretical interpretation is provided for constraint learning.

2) We design a novel pruning method to enhance the computational efficiency and reliability of constraint learning. As aforementioned, the conventional constraint learning may show undesirable computational efficiency because it has too many binary variable solutions. Nevertheless, many possible solutions can be removed a priori because their corresponding linear pieces i) do not contain feasible solutions or ii) lie in the area that the trained neural networks have not learned. The proposed pruning method identifies and removes these useless binary variable solutions in advance to reduce the computational burden. Moreover, this pruning method does not allow aggressive extrapolation because it restricts the solution close to training samples. As a result, the reliability of the neural network can also be enhanced.

The remaining parts are organized as follows. Section II describes the problem formulation of ADNs’ operation. Section III introduces the proposed PWL-based interpretation and pruning method. Section IV demonstrates simulation results, and Section V concludes this paper.

II. PROBLEM FORMULATION

We focus on a radial ADN with flexible resources. The operating goal is to minimize the total cost while satisfying all critical constraints. For simplicity, this paper uses HVAC systems as an example to represent flexible resources. Note that other flexible resources can also be considered in the same framework by adjusting the expression of nodal power injections. In this paper, we use non-Bold and Bold lowercase letters to represent scalars and vectors, while matrices are denoted by Bold uppercase letters.
A. System Modeling

1) HVAC systems: Due to buildings’ inherent ability to store heating/cooling power, the power profile of HVAC systems can be adjusted to promote the integration of DRG while keeping indoor thermal comforts. By using \( i \in \mathcal{I} \) and \( t \in \mathcal{T} \) to index HVAC systems and time slots, the thermodynamic model of one building can be expressed as [30]:

\[
\begin{align*}
\theta_{i,t}^{\text{in}} &= a_{i,t}^{\text{in}} \theta_{i,t-1}^{\text{in}} + a_{i,t}^{\text{out}} \theta_{i,t-1}^{\text{out}}, \\
\theta_{i,t}^{\text{out}} &= 1 - a_{i,t}^{\text{in}}, \\
\theta_{i,t}^{\text{h}} &= a_{i,t}^{\text{h}} (q_{i,t}^{\text{heat}} - q_{i,t}^{\text{cool}}) / g_i, \\
\end{align*}
\]

where \( \theta_{i,t}^{\text{in}} \) and \( \theta_{i,t}^{\text{out}} \) are the indoor and outdoor temperatures, respectively; \( q_{i,t}^{\text{heat}} \) and \( q_{i,t}^{\text{cool}} \) denote the heat load contributed by indoor sources (e.g., humans and electronic devices) and cooling supply of HVAC systems, respectively. Parameters \( a_{i,t}^{\text{in}}, a_{i,t}^{\text{out}}, \) and \( a_{i,t}^{\text{h}} \) are calculated by:

\[
a_{i,t}^{\text{in}} = e^{-\phi_i \Delta t}, \quad a_{i,t}^{\text{out}} = 1 - a_{i,t}^{\text{in}}, \quad a_{i,t}^{\text{h}} = a_{i,t}^{\text{out}} / g_i, \quad \forall i \in \mathcal{I}, \forall t \in \mathcal{T},
\]

where \( C_i \) and \( g_i \) are the building heat capacity and heat transfer coefficient between indoor and outdoor environments, respectively; \( \Delta t \) is the length of a time interval. Indoor thermal comforts require that all indoor temperatures should maintain in comfortable regions:

\[
\theta_{i,t}^{\text{min}} \leq \theta_{i,t} \leq \theta_{i,t}^{\text{max}}, \quad \forall t \in \mathcal{T},
\]

where \( \theta_{i,t}^{\text{min}} \) and \( \theta_{i,t}^{\text{max}} \) are the lower and upper bounds of comfortable regions, respectively. The active and reactive powers of HVAC systems are modeled by:

\[
\begin{align*}
\phi_i^H &= \frac{q_{i,t}^{\text{cool}}}{\text{COP}_{i,t}}, \\
\phi_i^H &= \frac{1 - a_{i,t}^{\text{in}}}{a_{i,t}^{\text{in}}} \phi_i^H, \quad \forall i \in \mathcal{I}, \forall t \in \mathcal{T},
\end{align*}
\]

where COP is the coefficient of performance of the \( i \)-th HVAC system; \( \phi_i \) is the power factor of the \( i \)-th HVAC system. The active power of HVAC systems should also be kept in allowable regions due to device limitations:

\[
P_{i,t}^H \leq P_{i,t}^{H,\text{max}}, \quad \forall t \in \mathcal{T},
\]

2) Distributed Renewable Generation: The actual outputs of distributed renewable generators \( P_{i}^{\text{DG}} \) can be controlled by adjusting their curtailment rate \( \lambda_i \), as follows:

\[
P_{i,t}^{\text{DG}} = G_i^{\text{DG}} \ast (1 - \lambda_i), \quad 0 \leq \lambda_i \leq 1, \quad \forall t \in \mathcal{T},
\]

where \( G_i^{\text{DG}} \) is the available DRG; operator \( \ast \) represents the element-wise multiplication.

3) Nodal Power Injections: The power injections on each node (except the root node), i.e., \( p_i \) and \( q_i \), are expressed as:

\[
p_i = -p_i^H - p_i^{\text{DG}} + p_i^d, \quad q_i = -q_i^H - q_i^{\text{DG}}, \quad \forall t \in \mathcal{T},
\]

where \( p_i^d \) and \( q_i^d \) are the base active and reactive demands, i.e., the demands except HVAC loads.

4) Power Flow Constraints: The power flow of a radial ADN can be described by the DistFlow model [31]:

\[
\begin{align*}
\sum_{k \in \mathcal{C}_i} P_{k,i,t} &= P_{j,i,t} - r_{ij} I_{ij,t}^2, \\
\sum_{k \in \mathcal{C}_i} Q_{k,i,t} &= Q_{j,i,t} - x_{ij} I_{ij,t}^2, \\
V_{j,t}^2 &= V_{i,t}^2 - 2 (r_{ij} P_{ij,t} + x_{ij} Q_{ij,t}) + (r_{ij}^2 + x_{ij}^2) I_{ij,t}^2, \\
I_{ij,t} &= \frac{P_{ij,t}^d + G_{ij,t}^d}{V_{i,t}}, \\
\forall (i,j) \in \mathcal{B}, \forall t \in \mathcal{T},
\end{align*}
\]

where \( P_{ij,t} \) and \( Q_{ij,t} \) are the active and reactive power flows on line \( (i,j) \), respectively; \( P_{ij,t} \) and \( Q_{ij,t} \) denote the active and reactive power injections on bus \( j \), respectively; \( V_{i,t} \) and \( I_{ij,t} \) are the voltage and current magnitudes on bus \( i \) and line \( (i,j) \), respectively. Symbol \( \mathcal{B} \) represents the index set of lines. Set \( \mathcal{C}_i \) contains the child bus indexes of bus \( j \). To guarantee security, the magnitudes of bus voltages and power flows shall stay in specific regions:

\[
V \leq V_t \leq \sqrt{P_t^2 + Q_t^2} = S_t \leq \overline{S}, \quad \forall t \in \mathcal{T},
\]

where \( V_t \) and \( S_t \) are the lower and upper bounds of voltage magnitudes; \( S_t \) and \( \overline{S} \) are the actual and maximum allowable values of apparent branch flows on lines.

5) Total Cost: The energy purchasing cost in one time slot, \( EC_t \), can be calculated by:

\[
EC_t = (\eta_{t,\text{buy}} G_{t,\text{buy}} - \eta_{t,\text{sell}} G_{t,\text{sell}}) \Delta t, \quad \forall t \in \mathcal{T},
\]

\[
G_{t,\text{buy}} - G_{t,\text{sell}} = G_{t,\text{root}}, \quad G_{t,\text{buy}} \geq 0, \quad G_{t,\text{sell}} \geq 0, \quad \forall t \in \mathcal{T},
\]

where \( \eta_{t,\text{buy}} \) and \( \eta_{t,\text{sell}} \) represent the per-unit prices of electricity purchasing and selling; \( \eta_{t,\text{buy}} \geq \eta_{t,\text{sell}} \); \( G_{t,\text{buy}} \) and \( G_{t,\text{sell}} \) are two auxiliary variables. Variable \( G_{t,\text{root}} \) is the net active power at the root node, which can be calculated by the network-level power balance:

\[
G_{t,\text{root}} = 1^T p_t + p_t^{\text{loss}}, \quad \forall t \in \mathcal{T},
\]

where \( p_t^{\text{loss}} \) is the total power loss and can be calculated by:

\[
p_t^{\text{loss}} = \sum_{(i,j) \in \mathcal{B}} I_{ij,t}^2 r_{ij}, \quad \forall t \in \mathcal{T}.
\]

B. Formulation of the Optimization Problem

The operating goal for an ADN is to minimize the energy purchasing cost while meeting all critical constraints, which can be summarized as \( P_1 \):

\[
\min_{(p_{i,t}^{H,\lambda_i})_{i,t \in \mathcal{T}}} \sum_{t \in \mathcal{T}} EC_t, \quad \text{s.t.: Eqs. (1)-(13).} \quad \text{(P1)}
\]

As mentioned in Section I, formulating \( P_1 \) is challenging because it requires network parameters, which may be unavailable in practice. Moreover, even if all parameters are known, \( P_1 \) is hard to solve due to the non-convex constraint (8).
III. SOLUTION METHODOLOGY

We propose an efficient constraint learning method to overcome the above challenges. In this section, we first introduce the conventional constraint learning in detail. Then, a PWL-based interpretation is proposed to explain why constraint learning can replicate constraints. Finally, a pruning method is developed to improve its computational efficiency.

A. Introduction of Constraint Learning

The key idea of constraint learning is to replace complex constraints with trained neural networks. Following reference [23], two MLPs are trained to replicate power flow constraints. The first MLP predicts constraint violations (termed as “Vio-MLP”), while the second one forecasts the total power loss (termed as “Loss-MLP”).

1) Vio-MLP: The input of the Vio-MLP, i.e., \( x_t \), is defined as the collection of the active/reactive power demands and the actually used DRG, as follows:

\[
x_t = (p_t^{HV} + p_t^{DG}, q_t^{HV} + q_t^{DG}, p_t^{DG}) \quad \forall t \in \mathcal{T}.
\]

The output of the Vio-MLP, i.e., \( h_t \), is defined as a measurement of power flow constraint violations, as follows:

\[
h_t = \max \left\{ \frac{V - V_i}{|V|}, \frac{V_i - V}{|V|}, \frac{S_t - S}{|S|} \right\} \quad \forall t \in \mathcal{T},
\]

where \( h_t \) is designed based on (9). Note that we standardize the violations of (9) to make the violations of voltage and power flow limitations comparable. Based on the operational data of the ADN, an MLP can be trained to describe the mapping from \( x_t \) to \( h_t \). In this paper, the Rectified Linear Unit (ReLU) is employed as the nonlinear activation function of MLPs. This activation function is defined as the positive part of its argument [32]. Then, the mapping from \( x_t \) to \( h_t \) can be approximated by the forward propagation of the trained MLP, as follows:

\[
s^0_t = x_t, \quad \forall t \in \mathcal{T},
\]

\[
s^l_t = W^l s^{l-1} + b^l, \quad \forall l \in \mathcal{L}, \quad \forall t \in \mathcal{T},
\]

\[
s^l_t = \max(z^l_t, 0), \quad \forall l \in \mathcal{L}, \quad \forall t \in \mathcal{T},
\]

\[
h_t = (w^{|L|+1})^\top s^{|L|} + b^{|L|+1}, \quad \forall t \in \mathcal{T},
\]

\[
\]

where \( z^l_t \) and \( s^l_t \) are the outputs of the linear mapping and activation function in hidden layer \( l \); \( \mathcal{L} \) denotes the set of hidden layers, \( l \in \mathcal{L} \); parameters \( (W^l, b^l)_{l \in \mathcal{L}} \) and \( (w^{|L|+1}, b^{|L|+1}) \) are the weights and bias that are to be learned through training.

By restricting the measurement \( h_t \) to be non-positive:

\[
h_t \leq 0, \quad \forall t \in \mathcal{T},
\]

power flow constraints can be satisfied. In other words, the non-convex power flow constraints (8)–(9) can be replicated by (16)–(20). Considering that the maximum operator in (18) is intractable for off-the-shelf solvers, reference [23] further equivalently reformulated constraints (17)–(18) into a solvable mixed-integer linear form, as follows:

\[
\begin{align*}
\{ s^l_t - r^l_t &= W^l s^{l-1} + b^l, \\
0 \leq s^l_t &\leq M \cdot \mu^l_t, \\
0 \leq r^l_t &\leq M \cdot (1 - \mu^l_t), \\
\mu^l_t &\in \{0, 1\} N_t,
\end{align*}
\]

where \( N_t \) denotes the neuron number in the \( l \)-th hidden layer. Obviously, each binary variable introduced in (21) represents the activation state of a neuron. Thus, the binary variable number introduced by the (21) is the same as the neuron number of the Vio-MLP.

2) Loss-MLP: According to (12), calculating the net power at the root node, \( G^0 \), requires the total power loss \( p^t_{\text{loss}} \). However, the value of \( p^t_{\text{loss}} \) is calculated based on the power flow model (8), which still requires network parameters. To address this issue, we train another MLP (recorded as “Loss-MLP”) to predict \( p^t_{\text{loss}} \). After training, the Loss-MLP can also be reformulated into a mixed-integer linear form. Since the forward propagation of the Loss-MLP is almost the same as that of the Vio-MLP, we record it as follows:

\[
\left\{ \text{Eqs. (16) and (21)} \right\}_{\text{loss}}.
\]

Here the superscript “loss” is employed to mark the parameters and variables used in the Loss-MLP.

Based on the above two MLPs and their mixed integer formulations, \( \textbf{P1} \) can be replicated by the following \( \textbf{P2}: \)

\[
\min_{(p^{HV}, \lambda t) \in \mathcal{T}} \sum_{t \in \mathcal{T}} E C_t,
\]

s.t. Eqs. (1)–(7), (10)–(12), (14), (16), and (19)–(22).

In \( \textbf{P2}, \) (1)–(7) and (14) describe the relationship between decision variables \( (p^{HV}, \lambda t) \) and input \( x_t \); (10)–(12) represent the calculation method of the energy cost \( E C_t \); (16), (19), and (21) denotes the Vio-MLP; (20) defines the safety condition; (22) represents the Loss-MLP.

Many published papers have shown that constraint learning has the potential to replicate power flow constraints without network parameters [21], [22], [23], [24]. However, this method still faces some challenges. Since neural networks are usually regarded as “black boxes,” it is hard to explain why constraint learning can replicate complex constraints. Meanwhile, it may also be computationally expensive due to the binary variables introduced in (21) and (22). Moreover, since this method trains MLPs to replicate constraints, its accuracy depends on the prediction performance of these MLPs. However, its solution may lie in an area that the MLPs have not learned. In this case, the prediction performance of MLPs can not be guaranteed, and the solution’s reliability may be hard to ensure.

B. Piecewise-Linearization-Based Interpretation

In this section, we show that constraint learning can be interpreted as a PWL technique for multivariate functions, which explains why it can learn complex constraints. Specifically, we
first introduce the concept “activation pattern.” It is a set that contains the activation states of all neurons in an MLP:

\[ \{ \mathbf{o}' \mid \forall l \in \mathcal{L} \}, \quad (23) \]

where vector \( \mathbf{o}' \) represents the activation states of all neurons in the \( l \)-th hidden layer. Its element is defined as:

1. If \( o'_{nl} = 1 \), then the \( n \)-th neuron in hidden layer \( l \) is active, and its ReLU’s input is non-negative.
2. If \( o'_{nl} = -1 \), then the \( n \)-th neuron in hidden layer \( l \) is inactive, and its ReLU’s input is negative.

Once the activation pattern is given, the positive/negative conditions can be regarded as constraints, an activation pattern also bounds a region \( \mathcal{R} \):

\[
\mathcal{R} = \left\{ \text{Activation states} \left| \text{diag}(\mathbf{o}') \begin{bmatrix} \hat{W}^l \mathbf{x}_t + \hat{b}^l \end{bmatrix} \geq 0, \forall l \in \mathcal{L} \right. \right\}, \quad (24)
\]

where \( \text{diag}(\mathbf{o}') \) denotes the diagonal matrix formed by the vector \( \mathbf{o}' \); \( \hat{W}^l \) and \( \hat{b}^l \) are calculated based on the learning parameters \( W^l \) and \( b^l \) in (16)–(19):

\[
\begin{align*}
\hat{W}^l &= W^l, \quad \hat{b}^l = b^l, \\
\tilde{W}^l &= \text{diag}(\mathbf{o}') \hat{W}^l, \quad \forall l \in \mathcal{L}/\{1\}, \\
\tilde{b}^l &= \text{diag}(\mathbf{o}') \hat{b}^l + b^l, \quad \forall l \in \mathcal{L}/\{1\}.
\end{align*}
\]

We call this \( \mathcal{R} \) as the “activation region” of the given activation pattern. Note that these activation regions are disjoint [33]. Then, in every activation region, the output \( h_t \) can be calculated based on a linear mapping of input \( \mathbf{x}_t \):

\[
h_t = (\tilde{w}^{[l]+1})^\top \mathbf{x}_t + \tilde{b}^{[l]+1}, \quad (26)
\]

where parameters \( \tilde{w}^{[l]+1} \) and \( \tilde{b}^{[l]+1} \) are defined as:

\[
\begin{align*}
\tilde{w}^{[l]+1} &= (w^{[l]+1})^\top \left( \text{diag}(\mathbf{o}') \tilde{W}^l \right), \\
\tilde{b}^{[l]+1} &= (w^{[l]+1})^\top \left( \text{diag}(\mathbf{o}') \tilde{b}^l \right) + b^{[l]+1}.
\end{align*}
\]

Note that each neuron has two activation states. Thus, one MLP can have \( |\mathcal{K}| = 2^{N_{\text{Total}}} \) activation regions, where \( N_{\text{Total}} \) is the total neuron number of this MLP.

Equation (20) requires that the measurement of constraint violations \( h_t \) should be non-positive. If we use \( k \in \mathcal{K} \) to index activation regions, then the feasible region in each activation region, i.e., \( \mathcal{F}_k \), can be expressed as:

\[
\mathcal{F}_k = \left\{ \mathbf{x}_t \left| \mathbf{x}_t \in \mathcal{R}_k, \quad h_{k,t} = (\tilde{w}_k^{[l]+1})^\top \mathbf{x}_t + \tilde{b}_k^{[l]+1} \leq 0 \right. \right\}, \forall k \in \mathcal{K}. \quad (28)
\]

Since all constraints in (28) is linear, set \( \mathcal{F}_k \) can be further expressed as a polytope:

\[
\mathcal{F}_k = \{ \mathbf{x}_t \mid A_k \mathbf{x}_t \leq \beta_k \}, \forall k \in \mathcal{K}, \quad (29)
\]

where \( A_k \) and \( \beta_k \) are the polytope’s parameters defined based on (28). Then, the feasible region formed by the conventional constraint learning, i.e., (16)–(20), can be rewritten as a union of \( \mathcal{F}_k \) in all activation regions:

\[
\mathbf{x}_t \in \bigcup_{k \in \mathcal{K}} \mathcal{F}_k. \quad (30)
\]

**Remark 1:** According to (23)–(30), the process of constraint learning can be summarized by Fig. 1. Based on different activation patterns, the trained MLP first partitions its input domain into disjoint linear pieces (activation regions). In each piece, the output of the MLP is linearly dependent on the input. Then, the feasible region of the original problem can be approximated by a union of polytopes. This manner is very similar to the PWL for multivariate functions.

**Remark 2:** Each activation region has a unique activation pattern. Meanwhile, when the activation pattern is given, the values of the binary variables in (21) are also determined. As a result, one activation region corresponds to a binary variable solution in constraint learning.

**Remark 3:** Increasing the number of linear pieces (i.e., activation regions) can reduce the approximation errors of the PWL (i.e., constraint learning). Meanwhile, reference [28] pointed out that the activation region number of an MLP is positively correlated with its neuron number. Thus, constraint learning can accurately replicate constraints once the MLP has a proper number of neurons.

We provide an example in Fig. 2 to further explain this PWL-based interpretation. In this example, constraint learning trains an MLP with three hidden layers to replicate a non-convex constraint \( y = x_1^2 - x_2^2 \geq 0.1 \) with \( x_1, x_2 \in [-1, 1] \). Each hidden layer contains five neurons. Fig. 2(a) shows the graph of the function \( y = x_1^2 - x_2^2 \). Fig. 2(b) illustrates different activation regions of the trained MLP, which are distinguished by colors. Fig. 2(c) demonstrates the output of the trained MLP. The output surface is composed of multiple planes. These planes demonstrate the linear mappings from \( x_1 \) and \( x_2 \) to \( y \). Fig. 2(d) compares the actual feasible region \( y \geq 0.1 \) (green areas) and the approximation provided by constraint learning (areas bounded by the red dash and black solid lines). Obviously, this feasible region can be accurately represented even if the MLP only has 15 neurons.

![Fig. 1. PWL-based interpretation of constraint learning.](image-url)
and feasible solutions simultaneously. Hence, we only need to compare at most 14 solutions after pruning. To identify all the unnecessary activation regions in our problem, we first check that each historical sample belongs to which activation region. Then, the activation regions that cover historical samples can be recognized. Next, the following optimization problem \( \text{P-S1} \) is solved to judge whether an activation region contains feasible solutions or not:
\[
\min_{x_t \in \mathcal{X}} \ 0, \quad \text{s.t. } x_t \in \mathcal{F}_k, \quad (\text{P-S1})
\]
where \( \mathcal{X} \) is the domain of nodal power injections \( x_t \), which can be approximated based on the historical data. If \( \text{P-S1} \) has solutions, then the \( k \)-th activation region contains feasible solutions of \( \text{P2} \). Otherwise, it does not contain any feasible solution and can be removed. Once all legal activation regions are found, constraint (30) can be replaced by:
\[
x_t \in \bigcup_{k \in \mathcal{K}^\text{Fe}} \mathcal{F}_k,
\]
where \( \mathcal{K}^\text{Fe} \) is the index set of the legal activation regions. Based on the Big-M method, constraint (31) can be further reformulated into a mixed-integer linear form:
\[
A_k x_{k,t} \leq z_{k,t} \beta_k, \quad \forall k \in \mathcal{K}^\text{Fe},
\]
\[
-M z_{k,t} \leq x_{k,t} - M z_{k,t}, \quad \forall k \in \mathcal{K}^\text{Fe},
\]
\[
\sum_{k \in \mathcal{K}^\text{Fe}} z_{k,t} = 1, \quad z_t \in \{0,1\}^{|\mathcal{K}^\text{Fe}|}.
\]
Obviously, due to the constraint \( \sum_{k \in \mathcal{K}^\text{Fe}} z_{k,t} = 1 \), at most \( |\mathcal{K}^\text{Fe}| \) binary variables are introduced in the worst case, which is much smaller compared to the conventional constraint learning, i.e., problem \( \text{P2} \).

2) Removing redundant constraints: According to (28), each polytope \( \mathcal{F}_k \) is bounded by multiple linear constraints. Since (32) involves multiple polytopes, it may introduce large-scale constraints and deteriorate the computational performance. To address this issue, we further design a step to remove redundant constraints for each polytope \( \mathcal{F}_k \) = \{ \( x_t \mid A_k x_t \leq \beta_k \) \}. Specifically, for a given polytope \( \mathcal{F}_k \), we can solve the following problem to identify whether its \( i \)-th constraint is redundant or not:
\[
\rho^{(i)}_k = \max_{x_t \in \mathcal{X}} A_k^{(i)} x_t,
\]
\[
\text{s.t. } A_k^{(m)} x_t \leq \beta_k^{(m)}, \forall m \in \mathcal{M}/\{i\}, \quad (\text{P-S2})
\]
where \( A_k^{(m)} \) and \( \beta_k^{(m)} \) represent the \( m \)-th rows of \( A_k \) and \( \beta_k \); \( m \in \mathcal{M} \) is the index of constraints in \( \mathcal{F}_k \). If the solution \( \rho^{(i)}_k \leq \rho^{(i)}_k \), then the \( i \)-th linear constraint, i.e., \( A_k^{(i)} x_t \leq \beta_k^{(i)} \), is redundant. Otherwise, it may be active.

It should be noted that removing one constraint may affect the redundancy of another constraint. For example, we may...
have $\rho^{(i)}_k \leq \beta^{(i)}_k$ and $\rho^{(j)}_k \leq \beta^{(j)}_k$ simultaneously for a given $\mathcal{F}_k$. It seems that both the $i$-th and $j$-th constraints are redundant. However, once the $i$-th constraint is removed, the $j$-th constraint may not be redundant anymore in the new polytope $\mathcal{F}'_k = \{ A^{(m)}_k x \leq \beta^{(m)}_k, \forall m \in \mathcal{M}/\{i\} \}$. Based on this observation, we design a heuristic algorithm to identify all redundant constraints, as shown in Fig. 3. Its detailed steps can be summarized as:

1) For a given polytope, solve $\text{P-S2}$ with $i \in \mathcal{M}$ in turn.
2) Once the first redundant constraint is found (indexed by $j$), remove it from $\mathcal{F}_k$, resulting in a new polytope.
3) Repeat Steps 1 and 2 until no redundant constraint can be found. Output the newest polytope (recorded as $\mathcal{F}'_k = \{ x | A^{pr}_k x \leq b^{pr}_k \}$).

Then, (32) can be equivalently simplified as:

$$A^{pr}_k x \leq z_k, b^{pr}_k, \forall k \in K^{Fe}. \quad (35)$$

D. Summary of the Efficient Constraint Learning

1) Final formulation: After applying the proposed two-step pruning, $\text{P2}$ can be replaced by:

$$\min_{(p_{3HV}, \lambda_k) \in \mathcal{T}} \sum_{t \in T} E C_t,$$

s.t. Eqs. (1)–(7), (10)–(12), (22), $\{(33) - (35)\}^{Vio}$. \quad (P3)

where symbol $\{(33) - (35)\}^{Vio}$ represents that the Vio-MLP is replicated by the efficient reformulation (33)–(35).\(^2\) Compared to $\text{P2}$, $\text{P3}$ introduces much fewer activation regions. In other words, fewer binary variable solutions need to be compared in B&B. Moreover, the redundant linear constraints are also dropped. Hence, its computational efficiency is significantly enhanced. Meanwhile, since the pruning method prevents excessive exploration, $\text{P3}$’s feasible region does not include those activation regions that the MLP has not learned. Hence, the reliability of the solutions can be also improved.

2) Merit of the proposed method: As introduced in Section I, the proposed method can bypass the requirement of network parameters by training neural networks based on historical samples. However, one can estimate the network parameters if the samples of all bus voltages, nodal injections, and branch flows are available. Then, model-based methods [6], [7], [8], can be employed because power flow equations can be established. Nevertheless, the proposed method still has merit. On the one hand, if only part of buses and branches are monitored, it is difficult to estimate all network parameters. In this case, those model-based methods are unable to apply because complete power flow equations are hard to build. Conversely, we can still calculate the measurement $h_t$ with those measured data based on (15). Then, the proposed method can derive a strategy that at least guarantees the security of the measured buses and branches. On the other hand, even if all samples are available, the proposed method can outperform those model-based methods in feasibility. Specifically, although power flow equations can be established in this case, they are non-convex. Thus, those model-based methods usually involve approximations or relaxations to address this non-convexity, which may introduce considerable errors and affect the feasibility performance. Instead, the proposed method trains MLPs to replicate power flow constraints without additional approximation or relaxation. Once these MLPs are accurate, the proposed method can achieve desirable feasibility. This advantage will also be verified via simulations in Sections IV-C and IV-D.

IV. CASE STUDY

A. Simulation Set Up

Two different case studies are implemented based on the IEEE 33- and 123-bus systems to verify the effectiveness of the proposed efficient constraint learning. In both cases, the time interval and optimization horizon are set as one hour and 24 hours, respectively. The parameters of buildings and HVAC systems are summarized in Table I. Other parameters will be introduced in Sections IV-C and IV-D.

The historical operational data of ADNs are simulated based on Pandapower, a power system simulation toolbox in Python [34]. Specifically, we first randomly generate 20,000 samples for the input $x_i$ with a uniform distribution. This uniform distribution is constructed based on the load profiles of standard IEEE
test systems and rated outputs of installed renewable generators, as follows:
\[ p_{i,t} = \kappa_{i}^{ac} \cdot p_{i}^{\text{stand}}, \quad \kappa_{i}^{ac} \sim U(0.5, 1.5), \quad \forall i \in \mathcal{I}, \]  
\[ q_{i,t} = \kappa_{i}^{ce} \cdot q_{i}^{\text{stand}}, \quad \kappa_{i}^{ce} \sim U(0.5, 1.5), \quad \forall i \in \mathcal{I}, \]  
\[ p_{i,t}^{DG} = \kappa_{i}^{DG} \cdot g_{i}^{\text{rated}}, \quad \kappa_{i}^{DG} \sim U(0, 1), \quad \forall i \in \mathcal{G}, \]  
where \( \kappa_{i}^{ac}, \kappa_{i}^{ce}, \) and \( \kappa_{i}^{DG} \) are random coefficients; \( p_{i}^{\text{stand}} \) and \( q_{i}^{\text{stand}} \) represent the active and reactive load on bus \( i \) in the standard IEEE test systems; \( g_{i}^{\text{rated}} \) is the rated output of the renewable generator on bus \( i \), and we set it as 3MW in this paper; \( \mathcal{I} \) is the index set of all buses and \( \mathcal{G} \) indexes the set of buses installed with distributed renewable generators. Note this sample generation manner has also been adopted in many other published works, such as [14], [35]. By giving these samples to Pandapower, bus voltages and power flows can be calculated. Based on (13) and (15), the power loss \( p_{i}^{\text{loss}} \) and measurement of the constraint violation \( h_{i} \) can be obtained. We gather all generated samples in the form of \( (x_{i}, h_{i}) \) and \( (x_{i}, p_{i}^{\text{loss}}) \) to construct training sets for the Vio-MLP and Loss-MLP, respectively. These samples have been uploaded in [36].

All numerical experiments are implemented on an Intel(R) 8700 3.20 GHz CPU with 16 GB memory. Our MLPs are implemented and trained by Pytorch. We employ CVXPY to build optimization problems and GUROBI to solve them.

B. Benchmarks

To demonstrate the benefits of the proposed method, two benchmarks are introduced:

1) B1: Original constraint learning used in [23], [24], [25]. It is a model-based method and does not involve any neural network.

2) B2: SOCP relaxation of the DistFlow used in [6], [7], [8].

3) B3: Binary classification-based constraint learning used in [21], [22] (termed “BC constraint learning”).

Note that the network parameters are unknown in the proposed method, B1 and B3, while it is known in B2.

C. Case Study Based on the IEEE 33-Bus System

1) Parameter Setting: The structure of the 33-bus system is shown in Fig. 4. It contains two distributed generators (marked by “DRG”). The voltage of its root node is 12.66 kV. The bus voltages are restricted in [0.9 p.u., 1.1 p.u.], which aligns with the default setting of the 33-bus test case in many well-recognized power system simulation tools, e.g., Pandapower [34] and Matpower [37]. The maximum allowable value of the line’s apparent branch flow is 4 MW. The heat transfer coefficient of the building, i.e., \( g_{i} \), is 0.03 MW/°C. The unit prices for electricity purchasing and selling, base power demands (the total demands except for HVAC loads), total heat loads contributed by indoor sources, and outdoor temperature are illustrated in Figs. 5(a)–(c). To better clarify the benefits of the proposed method, we also implement six scenarios with different amounts of available DRG, as shown in Fig. 5(d). To better clarify the benefits of the proposed method, we also implement six scenarios with different amounts of available DRG, as shown in Fig. 5(d).

2) Prediction accuracy of MLPs: As mentioned in Section III, the proposed method replicates power flow constraints based on trained regression MLPs. Hence, its accuracy depends on the prediction performance of these MLPs. Fig. 6 demonstrates the prediction accuracy of the MLPs trained in the proposed method, i.e., Vio-MLP and Loss-MLP, in the 33-bus case. Note we directly plot the actual and estimated values of the net power at the root node, i.e., \( G_{\text{root}}^{\text{net}} \), instead of the power loss here because the power loss is only used to estimate \( G_{\text{root}}^{\text{net}} \). The horizontal and vertical axes represent the actual values and corresponding estimations, while the red line is the position where the estimation equals the actual value. The structure of the Vio-MLP is (6, 6, 6), i.e., three hidden layers with six neurons in each layer. The structure of Loss-MLP is (3, 3, 3). Obviously, all the samples are pretty close to the corresponding red lines, which indicates the high prediction accuracy of MLPs. These results also imply the desirable accuracy of the proposed method.
TABLE II
COMPARISON BETWEEN THE CASES WITH AND WITHOUT PRUNING

| Numbers | Activation regions | Average constraints |
|---------|-------------------|---------------------|
| Before pruning | 262,144 | 19 |
| After pruning | 10 | 7.8 |

Fig. 7. Results of (a) total costs, (b) maximum voltage violations, and (c) maximum apparent branch flows obtained by different models in the 33-bus case. In (b), the y-axis is expressed in a logarithmic fashion.

3) Effectiveness of the Proposed Pruning Method: Table II compares the activation region numbers (column “Activation regions”) and average linear constraint numbers in each activation region (column “Average constraints”) before and after applying the proposed pruning method. Since the Vio-MLP contains 18 neurons, it totally has $2^{18} = 262,144$ activation regions. Meanwhile, according to (29), each activation region is formed by $18 + 1 = 19$ linear constraints. The first eighteen constraints form the activation region, i.e., $x_k \in R_{k_1}$, while the last one requires the violation measurement $h_{k, t}$ is non-positive. After simplification, the above two numbers decrease to 10 and 7.8, respectively. This demonstrates the effectiveness of our pruning method.

4) Optimality and Feasibility: Fig. 7 summarizes the total costs, maximum voltage violations, and maximum apparent branch flows of all models in the 33-bus test case. The six scenarios S1-S6 have different penetration of distributed renewable generation (DRG). From S1 to S6, the DRG penetration increases. In scenarios S1-S2 with small available DRG, there is no significant reverse power flow. Thus, the SOCP relaxation B2 only causes negligible constraint violations, and its solutions can be regarded as ideal ones. In scenarios S3-S6 with large available DRG, obvious reverse power flows occur in the system, so the SOCP relaxation may be inexact [11]. The conventional constraint learning B1 can achieve comparable optimality compared to B2 in scenarios S1-S2. Moreover, its feasibility is much better than that of B2. However, in scenario S1, its maximum violation of bus voltage and branch flow limitations reaches 0.02 p.u. and 0.57 MW, respectively. These violations are caused by the significant extrapolation from the training data, leading to an unreliable solution. The binary classification-based method B3 also causes obvious voltage and branch flow violations. Reference [23] pointed out that it is hard for B3 to accurately identify the feasible/infeasible boundary, so B3 may derive infeasible solutions. The total cost of the proposed method is almost the same as that in B1. The maximum relative cost difference between these two methods is always smaller than 0.2%. Moreover, the proposed method restricts the solution within the area that the Vio-MLP has learned. Thus, it can show better feasibility compared to B1. Among all scenarios, the voltage violation of the proposed method is always negligibly small. For example, this violation is only 0.0006 p.u. in scenario S4, which can be ignored in practice. Moreover, its maximum branch flow violation is 0.09 MW and significantly lower than that in B1. These results demonstrate the great optimality and feasibility of the proposed method.

We further compare the actual bus voltages of different methods to highlight the benefits of the proposed method in feasibility. Specifically, Fig. 8 presents the actual bus voltages in scenario S1. In the conventional constraint learning B1, the voltages on seven buses significantly violate the lower limit from 05:00 am to 10:00 am, and the maximum violation reaches 0.02 p.u. As aforementioned, B1 may extrapolate significantly from the training data, leading to an unreliable solution [29]. B2 is a relaxed model and may introduce relaxation error, resulting in constraint violations, e.g., four bus voltages exceed the lower...
Fig. 9. The bus voltages given by (a) the proposed method, (b) conventional constraint learning B1, (c) SOCP relaxation B2, and (d) Binary classification-based constraint learning B3.

Fig. 10. Solving times of different methods in the 33-bus case.

bound. Nevertheless, since the DRG penetration in S1 is relatively low, almost no reverse power flow occurs. Hence, this SOCP relaxation is almost exact [11], and the violations of B2 are very small. In the binary classification-based constraint learning B3, the voltages on six buses are lower than 0.9 p.u. from 7:00 am to 10:00 am. This is because it is hard for the binary classification-based method B3 to accurately identify the feasible/infeasible boundary [23]. Therefore, B3 may derive infeasible solutions. Unlike B1-B3, the proposed method can always keep all bus voltages within the allowable range [0.9 p.u., 1.1 p.u.] because it restricts the solution within the area close to training samples. Then, the prediction accuracy of neural networks can be guaranteed, and the feasibility of solutions can be ensured.

Fig. 9 illustrates the actual bus voltages of different methods in scenario S3. Both the proposed method and B1 can keep the voltage violations at a very low level, i.e., smaller than 0.001 p.u. On the contrary, the voltages of B2 significantly exceed this upper limit. In S6, reverse power flow occurs due to the high DRG penetration, so this SOCP relaxation becomes inexact and causes significant relaxation errors [11]. B3 also causes obvious voltage violations because it fails to accurately recognize the true position of feasible/infeasible boundary.

5) Computational Efficiency: Fig. 10 illustrates the solving times of different methods. The SOCP relaxation B2 shows excellent computational efficiency. However, it may perform poor feasibility according to Fig. 7. B3 also shows desirable computational performance, but it may cause significant violations. Although the proposed method spends more time than the conventional constraint learning B1 in scenarios S1-S3, its computational efficiency significantly outperforms B1 in the rest scenarios. Moreover, the proposed method can always keep the computational burden low, while B1 may show undesirable computational efficiency in some scenarios. For example, the solving process of the proposed method can be completed in 9.1 s in all scenarios, while it may take more than 111 s for B1 in S6. The proposed method involves a pruning method to remove unnecessary activation regions, which is equivalent to reducing the number of binary variable solutions for constraint learning. Hence, its computational complexity is lower than that of B1 in theory. However, besides the number of binary variable solutions, B&B’s computational complexity is also affected by many other characteristics (e.g., constraint number and parameter range), and its actual computation time for a specific problem is also influenced by initialization, branching heuristics, etc. [38], [39]. Thus, the proposed method may cost more time than B1 in some scenarios. Nevertheless, its average computational efficiency is significantly higher than that of B1 among all scenarios.

D. Case Study Based on the IEEE 123-Bus System

1) Parameter setting: Fig. 11 illustrates the structure of the 123-bus test system equipped with four distributed generators (marked as DRG). The voltage at the root node is 4.16 kV. The safe region of bus voltages are [0.9 p.u., 1.1 p.u.]. The unit prices for electricity purchasing/selling and outdoor temperature are the same as those of the 33-bus case. The heat transfer coefficient \( g_i \) is 0.02 MW/°C. The base electricity demand and indoor heat loads are shown in Fig. 12(a). Similarly, we also implement six scenarios with different available DRG, as demonstrated in Fig. 12(b).

2) Prediction Accuracy of MLPs: Fig. 6 shows the prediction accuracy of the Vio-MLP and Loss-MLP in the 123-bus case, where the structure of the two MLPs are (6, 6, 6) and (4, 4, 4), respectively.
respectively. Similarly, all sample points approach the red lines, which indicates the excellent performance of MLPs.

3) Optimality and feasibility: Fig. 14 summarizes the total costs, maximum voltage violations, and maximum apparent branch flows of all methods in different scenarios. Similar to the 33-bus case, the proposed method shows the best feasibility among all methods. For example, its maximum branch flow violation is only 0.07 MW, while this value reaches 0.41 MW, 2.21 MW, and 1.75 MW in B1, B2, and B3, respectively. The voltage violation of the proposed method is also maintained at a low level, i.e., smaller than 0.01 p.u. Moreover, the optimality of the proposed method is comparable to B1.

4) Computational efficiency: Fig. 15 demonstrates the solving time of different models. Both B2 and B3 show excellent computational efficiency. However, they may lead to infeasible solutions. Compared to B1, the proposed method takes more time in scenarios S1-S3 because it introduces additional linear constraints. Nevertheless, it demonstrates much better computational efficiency in scenarios S4-S6 since it removes many useless solutions of binary variables before solving. These results further confirm the desirable computational performance of the proposed method.

E. Robustness under Different Load/Generation Profiles

We have implemented Monte-Carlo simulations to verify the robustness of the proposed method under different load/generation profiles. These simulations are based on the IEEE 33-bus test system with 16 distributed renewable generators, as shown in Fig. 16. In the Monte-Carlo simulations, we test the performance of the proposed approach under 100 load/generation profiles (recorded as 100 different scenarios). Figs. 17(a)–(c) illustrate the active power demands, reactive power demands, and available DRG in these 100 scenarios.

We summarize the statistical results of the Monte-Carlo simulations to verify the effectiveness of the proposed method. Fig. 17(d)–(f) show the actual bus voltage and branch flow violations and the corresponding solving times. Although B2 shows the highest computational efficiency due to its convex property, its feasibility is undesirable. For example, constraint violations
to complete may be inexact, leading to huge relaxation errors and leads to significant constraint violations. Its maximum violations of bus voltages and branch flows reach 0.075 p.u. and 0.22 MW, respectively. As shown in Fig. 17(c), many scenarios have high DRG penetration, which may cause reverse power flows. In this case, the SOCP relaxation used in B2 may be inexact, leading to huge relaxation errors and poor feasibility performance [11]. The conventional constraint learning B1 also shows unacceptable feasibility. In most scenarios, B1 leads to significant constraint violations. Its maximum voltage and branch flow violations reach 0.033 p.u. and 0.53 MW, respectively. As aforementioned, B1 may extrapolate significantly from the training data, leading to an infeasible solution. Meanwhile, its computational efficiency is the worst one. For instance, it usually takes around 20 s for B1 to complete the solving, which is much longer compared to the rest two methods. Unlike B1 and B2, constraint violations appear in only a few scenarios when the proposed method is employed. Moreover, these constraint violations are significantly smaller than the other two methods. In addition, the solving time of the proposed method lies in [3 s, 8 s] in most scenarios, which is much less than that of B1. These results confirm the robustness of the proposed method under different load/generation profiles.

V. CONCLUSION

This paper proposes an efficient constraint learning method to operate ADNs. The proposed method replicates the power flow constraints by training two MLPs with no need for network parameters. A PWL-based interpretation is also presented to explain why the proposed method can accurately replicate complex constraints. Then, a pruning method is proposed to identify and drop the useless binary variable solutions before solving. Therefore, the computational burden of constraint learning can be maintained at a low level. Numerical experiments based on the IEEE 33- and 123-bus systems confirm that the proposed method can achieve better feasibility with comparable optimality compared to the conventional one. Moreover, its computational performance can always be guaranteed at a desirable level no matter how large the DRG penetration is.

This paper only considers the case with sufficient flexible resources so that the OPF is always feasible. However, in many practical systems, the controllable assets may be insufficient. In this case, neither the proposed method nor other scheduling methods can derive a feasible solution. Thus, we envision our future work to extend the proposed method to the case with no feasible OPF solution due to insufficient controllable assets.

REFERENCES

[1] P. J. Heptonstall and R. J. Gross, “A systematic review of the costs and impacts of integrating variable renewables into power grids,” Nature Energy, vol. 6, no. 1, pp. 72–83, 2021.
[2] X. Yang, C. Xu, H. He, W. Yao, J. Wen, and Y. Zhang, “Flexibility provisions in active distribution networks with uncertainties,” IEEE Trans. Sustain. Energy, vol. 12, no. 1, pp. 553–567, Jan. 2021.
[3] Y. Guo, Q. Wu, H. Guo, X. Chen, J. Østergaard, and H. Xin, “MPC-based coordinated voltage regulation for distribution networks with distributed generation and energy storage system,” IEEE Trans. Sustain. Energy, vol. 10, no. 4, pp. 1731–1739, Oct. 2019.
[4] K. Oikonomou, M. Parvania, and R. Khatami, “Deliverable energy flexibility scheduling for active distribution networks,” IEEE Trans. Smart Grid, vol. 11, no. 1, pp. 655–664, Jan. 2020.
[5] G. Chen, H. Zhang, H. Hui, and Y. Song, “Fast wasserstein-distance-based distributionally robust chance-constrained power dispatch for multi-zone HVAC systems,” IEEE Trans. Smart Grid, vol. 12, no. 5, pp. 4016–4028, Sep. 2021.
[6] J. Hu, X. Liu, M. Shahidehpour, and S. Xia, “Optimal operation of energy hubs with large-scale distributed energy resources for distribution network congestion management,” IEEE Trans. Sustain. Energy, vol. 12, no. 3, pp. 1755–1765, Jul. 2021.
[7] X. Kou et al., “A scalable and distributed algorithm for managing residential demand response programs using alternating direction method of multipliers (ADMM),” IEEE Trans. Smart Grid, vol. 11, no. 6, pp. 4871–4882, Nov. 2020.
[8] L. Lv, S. Chen, Z. Wei, and H. Zhang, “Power transportation coordination: Toward a hybrid economic-emission dispatch model,” IEEE Trans. Power Syst., vol. 37, no. 5, pp. 3969–3981, Sep. 2022.
[9] C. Wu, W. Gu, S. Zhou, and X. Chen, “Coordinated optimal power flow for integrated active distribution network and virtual power plants using decentralized algorithm,” IEEE Trans. Power Syst., vol. 36, no. 4, pp. 3541–3551, Jul. 2021.
[10] S. J. Pappu, N. Bhatt, R. Pasumarthy, and A. Rajeswaran, “Identifying topology of low voltage distribution networks based on smart meter data,” IEEE Trans. Smart Grid, vol. 9, no. 5, pp. 5113–5122, Sep. 2018.
[11] S. H. Low, “Convex relaxation of optimal power flow—Part II: Exactness,” IEEE Trans. Control. Netw. Syst., vol. 1, no. 2, pp. 177–189, Jun. 2014.
[12] Y. Wang, Q. Chen, T. Hong, and C. Kang, “Review of smart meter data analytics: Applications, methodologies, and challenges,” IEEE Trans. Smart Grid, vol. 10, no. 3, pp. 3125–3148, May 2019.
[13] G. Ruan, H. Zhong, G. Zhang, Y. He, X. Wang, and T. Pu, “Review of learning-assisted power system optimization,” CSEE J. Power Energy Syst., vol. 7, no. 2, pp. 221–231, 2021.
[14] X. Pan, T. Zhao, M. Chen, and S. Zhang, “DeepOPF: A deep neural network approach for security-constrained dc optimal power flow,” IEEE Trans. Power Syst., vol. 36, no. 3, pp. 1725–1735, May 2021.
[15] M. Chatzos, F. Fioletto, T. W. K. Mak, and P. Van Hentenryck, “High-fidelity machine learning approximations of large-scale optimal power flow,” 2020, arXiv:2006.16356.
[16] A. Velloso and P. Van Hentenryck, “Combining deep learning and optimization for preventive security-constrained DC optimal power flow,” IEEE Trans. Power Syst., vol. 36, no. 4, pp. 3618–3628, Jul. 2021.

[17] J. Arkhangelski, M. Abdou-Tankari, and G. Lefebvre, “Day-ahead optimal power flow for efficient energy management of urban microgrid,” IEEE Trans. Ind. Appl., vol. 57, no. 2, pp. 1285–1293, Mar./Apr. 2021.

[18] W. Wang, N. Yu, Y. Gao, and J. Shi, “Safe off-policy deep reinforcement learning algorithm for volt-var control in power distribution systems,” IEEE Trans. Smart Grid, vol. 11, no. 4, pp. 3008–3018, Jul. 2020.

[19] Y. Gao, W. Wang, and N. Yu, “Consensus multi-agent reinforcement learning for volt-var control in power distribution networks,” IEEE Trans. Smart Grid, vol. 12, no. 4, pp. 3594–3604, Jul. 2021.

[20] 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, Mar. 2022.

[21] A. Venzke and S. Chatzivasileiadis, “Verification of neural network behaviour: Formal guarantees for power system applications,” IEEE Trans. Smart Grid, vol. 12, no. 1, pp. 383–397, Jan. 2021.

[22] A. Venzke, G. Qu, S. Low, and S. Chatzivasileiadis, “Learning optimal power flow: Worst-case guarantees for neural networks,” in Proc. IEEE Int. Conf. Commun., Control, Comput. Technol. Smart Grids, 2020, pp. 1–7.

[23] G. Chen, H. Zhang, H. Hui, N. Dai, and Y. Song, “Scheduling thermostatically controlled loads to provide regulation capacity based on a learning-based optimal power flow model,” IEEE Trans. Sustain. Energy, vol. 12, no. 4, pp. 2459–2470, Oct. 2021.

[24] G. Chen, H. Zhang, H. Hui, and Y. Song, “Deep-quantile-regression-based surrogate model for joint chance-constrained optimal power flow with renewable generation,” IEEE Trans. Sustain. Energy, vol. 14, no. 1, pp. 657–672, Jan. 2023.

[25] B. Hanin and D. Rolnick, “Deep relu networks have surprisingly few activation patterns,” in Proc. Adv. Neural Inf. Process. Syst., 2019, vol. 32. [Online]. Available: https://proceedings.neurips.cc/paper/2019/file/97665272b5d3e595da4733cfb77b7e-Paper.pdf

[26] B. Grimstad and H. Andersson, “Relu networks as surrogate models in AC-OPF problems efficiently,” IEEE Trans. Power Syst., vol. 33, no. 6, pp. 6510–6521, Nov. 2018.

[27] A. A. Kadri, I. Kacem, and K. Labadi, “A branch-and-bound algorithm for solving the static rebalancing problem in bicycle-sharing systems,” Comput. Electric Eng., vol. 95, pp. 41–52, 2016.

[28] A. Venzke and S. Chatzivasileiadis, “Formal guarantees for power system applications,” IEEE Trans. Power Syst., vol. 36, no. 4, pp. 2423–2426. [Online]. Available: https://proceedings.neurips.cc/paper/2021/file/c7c403aa312160380010ee3dd48fc53-Paper.pdf

[29] G. Chen, H. Zhang, H. Hui, and Y. Song, “Deep-reinforcement learning-based optimal power flow model,” IEEE Trans. Sustain. Energy, vol. 11, no. 4, pp. 3008–3018, Jul. 2020.

[30] X. Chen, E. Dall’Anese, C. Zhao, and N. Li, “Aggregate power flexibility in unbalanced distribution systems,” IEEE Trans. Ind. Appl., vol. 56, no. 3, pp. 2601–2612, May/June 2020.

[31] H. Larochelle, A. Beygelzimer, F. D. Alché-Buc, E. Fox, and R. Garnett, “Learning to schedule heuristics in branch and bound,” in Proc. Adv. Neural Inf. Process. Syst., 2019, vol. 32. [Online]. Available: https://proceedings.neurips.cc/paper/2019/file/97665272b5d3e595da4733cfb77b7e-Paper.pdf

[32] B. Hanin and D. Rolnick, “Deep relu networks have surprisingly few activation patterns,” in Proc. Adv. Neural Inf. Process. Syst., 2019, vol. 32. [Online]. Available: https://proceedings.neurips.cc/paper/2019/file/97665272b5d3e595da4733cfb77b7e-Paper.pdf

[33] B. Hanin and D. Rolnick, “Deep relu networks have surprisingly few activation patterns,” in Proc. Adv. Neural Inf. Process. Syst., 2019, vol. 32. [Online]. Available: https://proceedings.neurips.cc/paper/2019/file/97665272b5d3e595da4733cfb77b7e-Paper.pdf

[34] B. Hanin and D. Rolnick, “Deep relu networks have surprisingly few activation patterns,” in Proc. Adv. Neural Inf. Process. Syst., 2019, vol. 32. [Online]. Available: https://proceedings.neurips.cc/paper/2019/file/97665272b5d3e595da4733cfb77b7e-Paper.pdf

[35] B. Hanin and D. Rolnick, “Deep relu networks have surprisingly few activation patterns,” in Proc. Adv. Neural Inf. Process. Syst., 2019, vol. 32. [Online]. Available: https://proceedings.neurips.cc/paper/2019/file/97665272b5d3e595da4733cfb77b7e-Paper.pdf

[36] B. Hanin and D. Rolnick, “Deep relu networks have surprisingly few activation patterns,” in Proc. Adv. Neural Inf. Process. Syst., 2019, vol. 32. [Online]. Available: https://proceedings.neurips.cc/paper/2019/file/97665272b5d3e595da4733cfb77b7e-Paper.pdf

[37] R. D. Zimmerman, C. E. Murillo-Sánchez, and R. J. Thomas, “MATPOWER: Steady-state operations, planning, and analysis tools for power systems research and education,” IEEE Trans. Power Syst., vol. 26, no. 1, pp. 12–19, Feb. 2011.

[38] A. A. Kadim, J. Kam, and K. Labadi, “A branch-and-bound algorithm for solving the static rebalancing problem in bicycle-sharing systems,” Comput. Electric Eng., vol. 95, pp. 41–52, 2016.

[39] A. Chiara, E. Khalil, A. Lodi, and S. Pokutta, “Learning to schedule heuristics in branch and bound,” in Proc. Adv. Neural Inf. Process. Syst., 2020, R. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, Eds., 2021, vol. 34, pp. 24223–24226. [Online]. Available: https://proceedings.neurips.cc/paper/2021/file/cb7c403aa312160380010ee3dd48fc53-Paper.pdf

Ge Chen (Graduate Student Member, IEEE) received the B.S. degree in thermodynamic engineering from the Huazhong University of Science and Technology, Wuhan, China, in 2015, and the M.S. degree in thermodynamic engineering from Xi’an Jiaotong University, Xi’an, China, in 2018. He is currently working toward the Ph.D. degree with the University of Macau, Zhuhai, China. His research interests include the Internet of Things for smart energy, optimal operation, and data-driven optimization under uncertainty.

Hongcai Zhang (Senior Member, IEEE) received the B.S. and Ph.D. degrees in electrical engineering from Tsinghua University, Beijing, China, in 2013 and 2018, respectively. He is currently an Assistant Professor with the State Key Laboratory of Internet of Things for Smart City, Department of Electrical and Computer Engineering, University of Macau, Zhuhai, China. During 2018–2019, he was a Postdoctoral Scholar with the Energy, Controls, and Applications Lab, University of California, Berkeley, CA, USA, where he was a Visiting Student Researcher in 2016. His research interests include the Internet of Things for smart energy, optimal operation and optimization of power and transportation systems, and grid integration of distributed energy resources.

Yonghua Song (Fellow, IEEE) received the B.E. and Ph.D. degrees in electrical engineering from the Chengdu University of Science and Technology, Chengdu, China, and the China Electric Power Research Institute, Beijing, China, in 1984 and 1989, respectively. He is currently a Chair Professor with the University of Macau, Zhuhai, China, and Director of the State Key Laboratory in the Internet of Things for Smart City. He is also the Vice President of Chinese Electrotechnical Society, an international Member of Academia Europaea, a Fellow of the Royal Academy of Engineering, U.K. He has long been engaged in renewable electrical power systems and smart energy research. He was the recipient of DSc by Brunel University, Uxbridge, U.K., in 2002, Honorary DEng by the University of Bath, Bath, U.K., in 2014, and Honorary DSc by The University of Edinburgh, Edinburgh, Scotland, in 2019. He was also the recipient of the second prize of the State Scientific and Technological Progress Award and the prize for Scientific and Technological Progress from the Ho Leung Ho Lee Foundation.