Interpretable Machine Learning for DC Optimal Power Flow with Feasibility Guarantees

Akylas Stratigakos, Salvador Pineda, Juan Miguel Morales, Georges Kariniotakis

Interpretable Machine Learning for DC Optimal Power Flow with Feasibility Guarantees

Akylas Stratigakos, Salvador Pineda, Juan Miguel Morales and Georges Kariniotakis

***Abstract*—The increased uncertainty due to the integration of stochastic renewable energy sources necessitates solving Optimal Power Flow (OPF) problems repeatedly and for a large number of scenarios. Machine learning methods hold significant potential to reduce the computing time for OPF problems by learning a mapping from varying input loads to decisions, thus bypassing the need for an optimization solver during inference. However, current machine learning methods for OPF lack interpretability and may produce infeasible decisions, which impedes their adoption by industry stakeholders. To this end, we propose a novel approach for interpretable learning of OPF solutions with feasibility guarantees. Specifically, we develop prescriptive decision trees that learn a piecewise affine mapping from input data to the solutions of a constrained optimization problem, using robust optimization to ensure that decisions are feasible. One important contribution of our work is the development of a tree-based learning method that utilizes hyperplane splits informed by domain knowledge, including network congestion and the merit order curve. By incorporating this information, our approach is able to enhance both the interpretability and performance of the model. We further present a surrogate learning algorithm to handle large-scale problems. The proposed approach is evaluated on several test networks, up to 300 buses, under different types of uncertainty and operating conditions, and is compared to neural network-based models, which do not guarantee feasibility. Notably, our results demonstrate that interpretable, shallow prescriptive trees perform comparably to neural network-based models, which are considered the current state of the art. To the best of our knowledge, this work is the first to introduce an interpretable machine learning approach for directly learning OPF solutions with guaranteed feasibility.**

***Index Terms*—Interpretable machine learning, DC Optimal Power Flow, decision trees, prescriptive analytics, robust optimization.**

I. Introduction

The optimal power flow (OPF) problem plays a crucial role in power system operation, planning, and electricity markets. It belongs to the class of network flow problems and its objective is to minimize the overall cost of power generation subject to power flow equations and operational constraints, e.g., transmission line limits. In various use cases, a linearized version of the OPF, referred to as DC-OPF [1], is utilized, especially in market clearing and contingency analysis scenarios. The DC-OPF problem is appealing as it can be expressed as a linear programming (LP) problem.

Although general-purpose optimization solvers have made solving LP problems efficient, certain settings can present computational challenges. With the increasing integration of renewable energy sources, significant uncertainty is introduced in both power supply and demand, resulting in the need to solve DC-OPF problems repeatedly and for a large number of scenarios. In this setting, traditional LP solvers may create a computational bottleneck.

Machine learning (ML) has been rapidly evolving in recent years, revolutionizing many industries, including power systems [2]. Due to their fast inference times, ML models have been proposed as an alternative to traditional optimization solvers for power system problems, such as the DC-OPF. However, power systems are critical infrastructure and stakeholders are naturally risk averse, which presents obstacles to the adoption of these tools [3]. Transparency, interpretability, and performance guarantees are necessary for practical implementation of ML models for problems such as the DC-OPF. Furthermore, interpretability should not compromise model performance but rather should be used to guide domain-agnostic methods with domain knowledge.

A. Literature Review

Leveraging ML to accelerate the solution of the DC-OPF problem has attracted significant attention in recent years. This work can be divided into two main research directions. The first focuses on end-to-end learning methods that directly predict the DC-OPF decisions, effectively emulating the LP solver. The second direction explores methods to find a reduced, and therefore easier to solve, DC-OPF problem.

The majority of research on end-to-end learning for DC-OPF focuses on utilizing neural network (NN) models to map uncertain load profiles to problem decisions [4]–[8]. For instance, [4] proposes a NN model with a constraint violation penalty to predict the DC-OPF solutions; a similar model is developed in [5] for security-constrained DC-OPF. To ensure feasibility of decisions, both models require a post-hoc projection step. In [6], worst-case constraint violations and suboptimality gap are estimated to verify the NN performance; a heuristic method to improve these worst-case guarantees by reducing the input domain is also proposed. In [7], physics-informed NNs demonstrate improved guarantees over standard NNs. However, ensuring that predicted decisions satisfy the

The work of A. Stratigakos and G. Kariniotakis was supported in part by the Smart4RES Project (Grant No 864337) funded under the Horizon 2020 Framework Program. The work of S. Pineda and J. M. Morales was supported in part by the European Research Council (ERC) funded under the Horizon 2020 Framework Program (Grant No 755705) and in part by the Spanish Ministry of Science and Innovation (AEI/10.13039/501100011033) through project PID2020-115460GB-I00.

A. Stratigakos and G. Kariniotakis are with Center for processes, renewable energy and energy systems (PERSEE), Mines Paris, PSL University, 06904 Sophia Antipolis, France: {akylas.stratigakos, georges.kariniotakis}@minesparis.psl.eu. S. Pineda and J. M. Morales are with the research group OASYS, University of Malaga, Malaga 29071, Spain: spineda@uma.es; juan.morales@uma.es.

problem constraints remains a challenge for end-to-end learning methods as prediction errors are inevitable. To address this issue, [8] develops a preventive learning framework to systematically calibrate inequality constraints to ensure feasibility; however, it relies on estimating the worst-case NN prediction error, which could be challenging. Nevertheless, even with feasible solutions guaranteed, NN models still lack the interpretability of other ML methods, such as decision trees [9]. Decision trees are inherently interpretable and have been used in power systems for decades — see, e.g., [10] for an early and [11] for a recent contribution on decision trees for dynamic security assessment.

Predicting the set of problem constraints that are binding at the optimal solution (*active set*) is generally simpler than directly predicting the optimal solutions. Motivated by this observation, the second research direction of leveraging ML for DC-OPF focuses on identifying the most probable active sets of constraints to find a reduced version of the original problem [12]–[16]. Specifically, [12] and [13] utilize statistical learning to identify the most probable critical regions, i.e., parameter regions where the active set of constraints remains unchanged, which then inform an ensemble policy. In [14], the problem of finding the active sets of constraints is formulated as a multiclass classification task. A neural decoding strategy is developed in [15] to first learn the active set of constraints, mapping uncertain load to the problem objective value, and then find solutions that satisfy the constraints. In the same line of work, [16] proposes a two-step process that combines prediction of active set of constraints with an iterative method to recover feasible solutions. While approaches based on learning the active sets of constraints are typically more interpretable and, in many cases, guarantee feasibility, they lack the inference speed of end-to-end learning. Nevertheless, this line of research offers a key insight: while the total number of active constraint sets is exponentially large, only a small number of them are relevant in practice.

In this work, we aim to reconcile these two research directions by proposing an end-to-end learning approach that combines the strengths of both methods and addresses their limitations. Drawing inspiration from recent progress in explainable prescriptive analytics [17], we leverage the insight that only a small number of active constraint sets are practically relevant to enhance both the performance and interpretability of our method. As such, rather than seeking a reduced DC-OPF problem, we develop an end-to-end learning method that is simpler in complexity.

It is worth noting that multiparametric programming [18] is another research area relevant to leveraging ML for DC-OPF. Multiparametric programming aims to solve constrained optimization problems as a function of uncertain parameters by identifying critical regions and explicitly constructing a parameter-dependent solution for the whole parameter space. The key difference from our work is that we do not aim to explore the whole parameter space but rather derive an interpretable policy that encodes a few key rules selected in a data-driven manner, starting from available data, and ensuring feasibility for the whole (unobserved) parameter space.

B. Aim and contribution

In this paper, we present a novel method for affine prescriptive trees, i.e., decision trees that learn a piecewise affine mapping from varying input data to the solutions of a constrained optimization problem, namely the DC-OPF problem. We develop a new learning algorithm that combines parallel and *domain-informed* hyperplane splits that encode network information, namely the merit order curve and network congestion. We formulate the expectation of network congestion, conditioned on load, as a classification task and model it with support vector machine (SVM) classifiers. The separating hyperplanes derived from the SVM models are then used as input in the tree learning algorithm, simultaneously improving model performance and interpretability. We also use robust optimization to ensure feasibility of the predicted decisions for the whole parameter space in a principled manner. A surrogate learning algorithm is also developed to address the case of potentially prohibitive training time for large-scale problem instances. We provide comprehensive numerical experiments for several test cases ranging from 5 to 300 bus systems, under different assumptions for the distribution of uncertainty and operating conditions. The results show that our method achieves similar performance with state-of-the-art end-to-end learning approaches, namely neural network-based models, while also maintaining interpretability and ensuring feasibility of decisions.

In summary, our main contribution is twofold. Firstly, we propose an interpretable end-to-end learning method for DC-OPF that offers fast solutions during inference, is computationally tractable, and provides feasibility guarantees. Secondly, we propose a two-step process to learn decision trees with hyperplane splits that encode domain-specific information, thereby improving performance and retaining interpretability. To the best of our knowledge, our work is the first to develop interpretable end-to-end ML for the DC-OPF problem with feasibility guarantees.

C. Paper Organization

The remainder of the paper is organized as follows. Section II formulates the problem of learning DC-OPF solutions. Section III develops the tree-based methodology. Section IV illustrates the proposed methodology in a small test case. Section V presents our numerical experiments. Section VI concludes and provides directions for future work.

II. DC-OPF and Learning Problem Formulation

This section introduces the DC-OPF problem (Subsection II-A), describes the proposed learning problem (Subsection II-B), and illustrates how to reformulate it into a tractable problem (Subsection II-C).

A. The DC-OPF Problem

This section formulates the DC-OPF problem. Throughout, boldfaced lowercase letters denote vectors, boldfaced uppercase letters denote matrices, calligraphic font denotes sets, and $|\cdot|$ denotes the set cardinality. We consider a transmission

network where $\mathcal{V}$ is the set of buses, $\mathcal{E}$ is the set of lines, and $\mathcal{G}$ is the set of generators.

The deterministic DC-OPF problem writes

$$\min_{\mathbf{p}} \quad \mathbf{c}^\intercal \mathbf{p}, \tag{1a}$$

$$\text{s.t.} \quad \mathbf{1}^\intercal \mathbf{p} - \mathbf{1}^\intercal \mathbf{d} = 0, \tag{1b}$$

$$-\overline{\mathbf{f}} \leq \mathbf{M}(\mathbf{A}\mathbf{p} - \mathbf{d}) \leq \overline{\mathbf{f}}, \tag{1c}$$

$$\mathbf{0} \leq \mathbf{p} \leq \overline{\mathbf{p}}, \tag{1d}$$

where $\mathbf{p} \in \mathbb{R}^{|\mathcal{G}|}$ denotes the active power of dispatchable generators, $\mathbf{d} \in \mathbb{R}^{|\mathcal{V}|}$ is the stochastic net demand (load demand minus renewable generation) at each bus, $\mathbf{M} \in \mathbb{R}^{|\mathcal{E}|\times|\mathcal{V}|}$ is the PTDF matrix, $\mathbf{A} \in \mathbb{R}^{|\mathcal{G}|\times|\mathcal{V}|}$ is an incidence matrix mapping generators to buses, and $\mathbf{1}(\mathbf{0})$ is a vector of ones (zeros) with appropriate size. Further, $\mathbf{c}, \overline{\mathbf{p}}$, and $\overline{\mathbf{f}}$ are known positive parameters that define the generation cost, the generator capacity, and the line capacity, respectively. The problem objective (1a) minimizes the total generation cost, (1b) ensures balance of demand and supply, while (1c) and (1d) denote the generation and transmission line limits, respectively. Without loss of generality we assume a linear cost function in the objective; quadratic cost functions can always be approximated by a piecewise linear function.

Next, we present some assumptions that apply in this work regarding the DC-OPF problem (1).

Assumption 1: The net load $\mathbf{d}$ is restricted in the polyhedron

$$\mathcal{U} = \{\mathbf{d} \in \mathbb{R}^{|\mathcal{V}|} \,|\, \mathbf{H}\mathbf{d} \leq \mathbf{h}\}. \tag{2}$$

This is a standard assumption. In practice, the net load at each bus may vary within a pre-specified range. Formally, this is defined as

$$\mathcal{A} = \{\mathbf{d} \in \mathbb{R}^{|\mathcal{V}|} \,|\, \underline{\mathbf{d}} \leq \mathbf{d} \leq \overline{\mathbf{d}}\}, \tag{3}$$

where $\underline{\mathbf{d}}\,(\overline{\mathbf{d}})$ denotes the minimum (maximum) values, with renewable production being defined with negative values. Observe that (3) is a special case of (2), where $\mathbf{H} = [\mathbf{I}, -\mathbf{I}]^\intercal$ and $\mathbf{h} = [\overline{\mathbf{d}}, \underline{\mathbf{d}}]^\intercal$, where $\mathbf{I}$ denotes an identity matrix of appropriate size.

Assumption 2 (Feasibility): Problem (1) is feasible $\forall \mathbf{d} \in \mathcal{U}$.

Note that if the deterministic formulation of the DC-OPF problem is infeasible, then slack variables need to be included in (1). For simplicity, we assume that (1) is always feasible; however, our proposed method can be straightforwardly extended to address the case when slack variables are required.

Assumption 3 (Uniqueness): Problem (1) admits a unique solution $\forall \mathbf{d} \in \mathcal{U}$.

This is also a standard assumption, which holds almost surely for appropriate cost vectors [19].

B. Data-driven Piecewise Affine Policy

This section presents the proposed data-driven piecewise affine policy for end-to-end learning of the DC-OPF problem. First, we define a polyhedral partition of the feasible space $\mathcal{U}$.

Definition 1 (Polyhedral partition [20]): A collection of L polyhedra $\{\mathcal{U}_\ell\}_{\ell=1}^L$ is a polyhedral partition of a set $\mathcal{U}$ if $\mathcal{U} = \cup_{\ell=1}^L \mathcal{U}_\ell$ and $(\mathcal{U}_i \setminus \partial\mathcal{U}_i) \cap (\mathcal{U}_j \setminus \partial\mathcal{U}_j) = \emptyset$, $\forall i \neq j$, where $\partial\mathcal{U}_i$ denotes the boundary of $\mathcal{U}_i$ and $\setminus$ denotes the set difference operator. In other words, the union of the individual polyhedra $\mathcal{U}_\ell$ covers the feasible space of the net load, and the interiors of the polyhedra do not overlap.

In this work, we assume that a data set $\mathcal{D} = \{(\mathbf{d}_i, \mathbf{p}_i^*)\}_{i=1}^N$ of N training observations is available, where $\mathbf{d}_i$ denotes the net load and $\mathbf{p}_i^*$ denotes the vector of optimal decisions derived from solving (1) for the i-th sample. In a data-driven setting, a polyhedral partition $\{\mathcal{U}_\ell\}_{\ell=1}^L$ also implies a respective partition of training data $\{\mathcal{D}_\ell\}_{\ell=1}^L$, i.e., subsets of data that fall in each polyhedron. Formally, we define

$$\mathcal{D}_\ell := \{(\mathbf{d}_i, \mathbf{p}_i^*), i \in [N] \,|\, \mathbf{d}_i \in \mathcal{U}_\ell\} \subseteq \mathcal{D}, \tag{4}$$

where $[N]$ is shorthand for $\{1, \ldots, N\}$.

In the following, we present the proposed data-driven piecewise affine policy that maps net load observations to decisions. First, we particularize Definition 1 to the current data-driven setting.

Definition 2 (N_{min}-admissible polyhedral partition): Consider a scalar $N_{min} > 0$, a polyhedral partition $\{\mathcal{U}_\ell\}_{\ell=1}^L$, and a corresponding data partition $\{\mathcal{D}_\ell\}_{\ell=1}^L$. We say that $\{\mathcal{U}_\ell\}_{\ell=1}^L$ is N_{min}-admissible, if $|\mathcal{D}_\ell| \geq N_{min}$, $\forall \ell \in [L]$.

Therefore, Definition 2 only considers polyhedral partitions where each polyhedron includes a minimum number of data observations; here is also where our approach differentiates from multiparametric programming [18]. The utility of Definition 2 will become apparent in the next section, where we develop the tree-learning algorithm. In that case, N_{min} corresponds to the minimum number of observations per each tree leaf.

The proposed data-driven piecewise affine policy is defined as follows.

Definition 3 (Data-driven piecewise affine policy): We consider a data-driven piecewise affine policy $f : \mathcal{U} \to \mathbb{R}^{|\mathcal{G}|}$ that maps net load $\mathbf{d}$ to generator setpoints $\mathbf{p}$, given by $f(\mathbf{d}) = \mathbf{W}_\ell \mathbf{d} + \mathbf{b}_\ell$, $\mathbf{d} \in \mathcal{U}_\ell$, $\ell = 1, \ldots, L$, where $\mathbf{W}_\ell \in \mathbb{R}^{|\mathcal{G}|\times|\mathcal{V}|}$ is a matrix of linear decision rules, $\mathbf{b}_\ell$ is the intercept vector, and $\{\mathcal{U}_\ell\}_{\ell=1}^L$ is an N_{min}-admissible polyhedral partition of $\mathcal{U}$, defined over a data set $\mathcal{D}$.

Given an N_{min}-admissible polyhedral partition $\{\mathcal{U}_\ell\}_{\ell=1}^L$, the problem of finding the optimal decision rules, for each $\ell \in [L]$, is given by

$$\min_{\mathbf{W}_\ell, \mathbf{b}_\ell} \quad \frac{1}{|\mathcal{D}_\ell|} \sum_{i \in \mathcal{D}_\ell} \mathbf{c}^\intercal (\mathbf{W}_\ell \mathbf{d}_i + \mathbf{b}_\ell), \tag{5a}$$

$$\text{s.t.} \quad \mathbf{1}^\intercal (\mathbf{W}_\ell \mathbf{d} + \mathbf{b}_\ell) - \mathbf{1}^\intercal \mathbf{d} \geq 0, \qquad \forall \mathbf{d} \in \mathcal{U}_\ell, \tag{5b}$$

$$-\overline{\mathbf{f}} \leq (\mathbf{A}(\mathbf{W}_\ell \mathbf{d} + \mathbf{b}_\ell) - \mathbf{d}) \leq \overline{\mathbf{f}}, \quad \forall \mathbf{d} \in \mathcal{U}_\ell, \tag{5c}$$

$$\mathbf{0} \leq \mathbf{W}_\ell \mathbf{d} + \mathbf{b}_\ell \leq \overline{\mathbf{p}}, \qquad \forall \mathbf{d} \in \mathcal{U}_\ell, \tag{5d}$$

where the decision vector $\mathbf{p}$ has been replaced by the affine policy $\mathbf{W}_\ell \mathbf{d} + \mathbf{b}_\ell$. Problem (5) finds the affine decision rules that minimize the in-sample dispatch cost (5a) for the given partition. Note that each row of $\mathbf{W}_\ell$ defines a vector of coefficients that maps net load to a specific generator. The robust constraints (5b)-(5d) further ensure a feasible policy, i.e., decisions are feasible for all realizations of the uncertainty within $\mathcal{U}_\ell$. At inference, for an out-of-sample observation $\mathbf{d}_0$, we first locate the respective partition $\mathcal{U}_\ell$ it falls into, and then derive the generator production from $f(\mathbf{d}_0) = \mathbf{W}_\ell \mathbf{d}_0 + \mathbf{b}_\ell$.

Note that in the robust formulation we replaced the equality constraint (1b) with an inequality constraint (5b); thus, problem (5) ensures that total net load is always covered by the aggregated production. The reason for this is twofold. First, equality constraints with uncertain parameters drastically reduce the feasible set, leading to over-conservative solutions or even infeasibility [21, Ch. 12]. Second, in reality, generation must always be larger than demand due to line losses. Moreover, there are always small deviations between aggregated production and net load, which result in frequency variations; as it is easier to provide downward frequency regulation via, e.g., curtailment of renewable production, we ensure that the policy never underestimates the total demand.

Remark 1: If Assumption 2 does not hold, then (1) requires additional slack variables. In this case, we introduce additional rules in (5) that map realizations of $\mathbf{d}$ to each slack variable.

The objective (5a) minimizes the prescriptive cost, i.e., the expected in-sample dispatch cost. Alternatively, the mean squared error (MSE) between the optimal and forecast decisions can be minimized, given by

$$\frac{1}{|\mathcal{D}_\ell|}\sum_{i\in\mathcal{D}_\ell}\|\mathbf{W}_\ell\mathbf{d}_i+\mathbf{b}_\ell-\mathbf{p}_i^*)\|_2^2, \tag{6}$$

as is the case in many relevant works — see, e.g., [6]. The MSE measures the predictive error of forecast decisions. However, here we focus exclusively on the prescriptive cost, as the ultimate goal is to minimize the total dispatch cost.

C. Robust Constraint Reformulation

Problem (5) involves semi-infinite robust constraints. As we deal with an LP problem and polyhedral uncertainty sets, we apply techniques from robust optimization [21] to reformulate (5) into a deterministic LP.

For illustration purposes, consider the upper generation limit at the left-hand side (*l.h.s.*) of (5d). Considering that the inequality holds $\forall\mathbf{d}\in\mathcal{U}_\ell$, i.e., the worst-case of $\mathbf{d}$, we write equivalently

$$\max_{\mathbf{d}}\{\mathbf{W}_\ell\mathbf{d}\,|\,\mathbf{H}_\ell\mathbf{d}\le\mathbf{h}_\ell\}\le\overline{\mathbf{p}}-\mathbf{b}_\ell.$$

As the $\max$ problem is linear in $\mathbf{d}$, it can be replaced by its dual

$$\min_{\boldsymbol{\lambda}}\{\mathbf{h}_\ell^\intercal\boldsymbol{\lambda}\,|\,\mathbf{H}_\ell^\intercal\boldsymbol{\lambda}=\mathbf{W}_\ell,\boldsymbol{\lambda}\ge\mathbf{0}\}\le\overline{\mathbf{p}}-\mathbf{b}_\ell,$$

where $\boldsymbol{\lambda}$ is a dual variable of appropriate size. Evidently, the $\min$ operator becomes redundant. Hence, the upper generation limit constraint in the *l.h.s.* of (5d) is replaced by the following constraints

$$\mathbf{h}_\ell^\intercal\boldsymbol{\lambda}\le\overline{\mathbf{p}}-\mathbf{b}_\ell,\ \mathbf{H}_\ell^\intercal\boldsymbol{\lambda}=\mathbf{W}_\ell,\ \boldsymbol{\lambda}\ge\mathbf{0}.$$

The rest of the constraints are reformulated in a similar fashion, leading to a deterministic LP problem that can be solved with off-the-shelf solvers.

III. Tree-based Learning Methodology

This section develops the proposed tree-based method to learn an interpretable policy for the DC-OPF problem. First, we describe the tree-learning algorithm (Subsection III-A). Next, we detail the process of finding domain-informed hyperplane splits (Subsection III-B). Finally, we describe a surrogate learning method to deal with large problem instances (Subsection III-C).

A. Affine Prescriptive Trees

In this section, we present our decision tree algorithm for learning a piecewise affine policy from a data set $\mathcal{D}$. Decision trees use available data to partition the feature space into L leaves by minimizing a predefined loss criterion, e.g., minimizing the variance of each leaf. The resulting partition also provides information about the joint distribution of the target variable and associated features, therefore can be used to predict instances of the target variable given out-of-sample feature observations.

Our proposed algorithm combines parallel and hyperplane splits during the tree learning process, which is a departure from most state-of-the-art tree algorithms that focus solely on binary trees with parallel splits— see, e.g., [22] for single trees and [23] for tree-based ensembles. Oblique decision trees [24] allow for hyperplance splits and have been shown to lead to significant performance improvements; however, they can be computationally challenging and less interpretable [9]. To address this challenge, we construct a set of domain-informed hyperplane splits prior to the learning phase; the process of identifying these splits is detailed in the next section.

Algorithm 1 describes our decision tree algorithm in detail. Consider a root node (equivalently, partition) $\mathcal{U}_0=\{\mathbf{d}\,|\,\mathbf{H}_0\mathbf{d}\le\mathbf{h}_0\}$, a corresponding data set $\mathcal{D}_0$, and a set of K candidate hyperplane splits $\{(\boldsymbol{\alpha}_k,\beta_k)\}$, parameterized by vectors $\boldsymbol{\alpha}_k$ and scalars β_k. Note that parallel splits are a special case, e.g., if we want to split in value s of feature d_1, then $\boldsymbol{\alpha}_k=[1,\mathbf{0}]^\intercal$, and $\beta_k=s$.

A node split partitions a parent node into two child nodes $\mathcal{U}_0=\mathcal{U}_l\cup\mathcal{U}_r$, such that $\mathcal{U}_l=\{\mathbf{d}\,|\,\boldsymbol{\alpha}_k^\intercal\mathbf{d}\le\beta_k,\,\mathbf{d}\in\mathcal{U}_0\}$ and $\mathcal{U}_r=\{\mathbf{d}\,|\,\boldsymbol{\alpha}_k^\intercal\mathbf{d}>\beta_k,\,\mathbf{d}\in\mathcal{U}_0\}$. The training algorithm starts at root node $\mathcal{U}_0$ and sets the current depth $\delta=0$. Next, it iterates over the K candidate splits and solves (5) for each child partition; note that, to deal with the strict inequality induced by the node split, the right child node is evaluated at its closure $\text{cl}(\mathcal{U}_r)$. The split that minimizes the prescriptive cost of the piecewise affine policy is selected and the corresponding polyhedral partition is added to the tree, updating the tree structure accordingly. At each iteration, the current tree leaves define an N_{min}-admissible polyhedral partition $\{\mathcal{U}_\ell\}_{\ell=1}^L$ and an equivalent data partition $\{\mathcal{D}_\ell\}_{\ell=1}^L$. The process is repeated recursively in a top-down fashion until a stopping criterion is met. Typical stopping criteria include a minimum number of observations per leaf N_{min} and the maximum tree depth δ^{max}.

The proposed tree-learning algorithm grows trees that minimize decision cost and map data to prescriptions. We take

an intermediate approach to split selection, avoiding the well-known shortcoming of CART-like methods [22], which is determining each split without considering the possible impact on future splits[1]. Specifically, we apply a semi-greedy split selection, which prioritizes hyperplane splits over parallel ones, as the former encode domain knowledge. To implement the semi-greedy split selection, we use an auxiliary function called `ispar`, which takes a vector $\boldsymbol{\alpha}_k$ as input and returns a logical value of `True` if $\boldsymbol{\alpha}_k$ is parallel and `False` otherwise. In the tree-learning algorithm, if the current best split is non-parallel, we only evaluate the remaining hyperplane splits. If the current best split is parallel, then we evaluate all the remaining splits, including parallel ones. This is described in Steps 4-5 of Algorithm 1, where $\neg, \wedge$ denote the logical negation and conjuction (*and*) operators, and the **continue** statement interrupts the current step of a loop and continues with the next iteration. This approach prioritizes domain-informed hyperplane splits while still allowing for data-driven parallel splits to be considered if the hyperplane splits are insufficient.

The hyperparameters of the decision tree include the minimum number of observations N_{min} per leaf and the maximum tree depth δ^{max}. The former controls the bias-variance trade-off, with smaller values increasing the risk of overfitting. Additionally, N_{min} ensures that the final polyhedral partition is admissible as per Definition 2. Conversely, larger values of δ^{max} lead to improved performance, but may also results in overfitting and reduced interpretability. Therefore, to promote model interpretability, we suggest using larger values of N_{min} and smaller values of δ^{max}.

Algorithm 1 `AffinePrescrTree`

Input: current partition $\mathcal{U}_0$, current data set $\mathcal{D}_0$, current depth δ, hyperparameters $\{N_{min}, \delta^{max}\}$, set of candidate splits $\{(\boldsymbol{\alpha}_k, \beta_k)\}_{k=1}^K$, auxiliary function `ispar`

Output: tree T

1: find $v_0 = \min_{\mathbf{d}\in\mathcal{U}_0}$ (5), set $v_{min} \leftarrow |\mathcal{D}_0| v_0$, `split`$\leftarrow$`False`, $k^* \leftarrow$ empty
2: **if** $\delta < \delta^{max}$ and $N_0 \geq 2N_{min}$ **then**
3: **for** $k = 1, \ldots, K$ **do**
4: **if** $\neg$`ispar`$(\boldsymbol{\alpha}_{k^*}) \wedge$ `ispar`$(\boldsymbol{\alpha}_k) ==$ `True` **then**
5: **continue**
6: **else**
7: find left and right child nodes $\mathcal{U}_l, \mathcal{U}_r$, and corresponding data partitions $\mathcal{D}_l, \mathcal{D}_r$
8: **if** $|\mathcal{D}_l| \geq N_{min}$ and $|\mathcal{D}_r| \geq N_{min}$ **then**
9: $v_k = |\mathcal{D}_l| \cdot \min_{\mathbf{d}\in\mathcal{U}_l}$ (5) $+ |\mathcal{D}_r| \cdot \min_{\mathbf{d}\in\text{cl}(\mathcal{U}_r)}$ (5)
10: **if** $v_k < v_{min}$ **then**
11: update $v_{min} \leftarrow v_k$, `split` $\leftarrow$ `True`, $k^* \leftarrow k$
12: **end if**
13: **end if**
14: **end if**
15: **end for**
16: **if** `split == True` **then**
17: append $(\boldsymbol{\alpha}_{k^*}, \beta_{k^*})$ to $\mathbf{H}_0, \mathbf{h}_0$ for each new partition $\mathcal{U}_l, \mathcal{U}_r$, find $\mathcal{D}_l, \mathcal{D}_r$
18: $T_l =$ `AffinePrescrTree`$(\mathcal{U}_l, \mathcal{D}_l, \delta + 1)$
19: $T_r =$ `AffinePrescrTree`$(\mathcal{U}_r, \mathcal{D}_r, \delta + 1)$
20: update tree structure T
21: **end if**
22: **end if**
23: **return** T

B. Domain-Informed Hyperplane Splits

This section describes how to identify the set of K candidate splits.

Parallel splits are splits that only check whether an entry of $\mathbf{d}$ exceeds a threshold value; they are purely data-driven and the standard approach to grow binary trees, e.g., CART. In this work, the set of parallel splits comprises a number of equally spaced quantiles of the empirical net load distribution over data set $\mathcal{D}$; i.e., for each net load at each node we estimate a set of quantiles from its marginal distribution and evaluate the splitting criterion there.

A key contribution of this work is proposing domain-informed hyperplane splits that are potentially more effective than data-driven parallel splits. The proposed hyperplane splits encode information about the active set of constraints conditioned on the load profile, namely the merit order curve and network congestion.

1) Merit order splits: For ease of discussion, further assume the generators in $\mathcal{G}$ are ordered in ascending order based on their cost, i.e., for $i, j \in \mathcal{G}$, if $i < j$, then $c_i < c_j$. Hence, for an optimal solution $\mathbf{p}^*$, assuming no line congestion, we have $p_i = \overline{p}_i$ whenever $p_j > 0$. This means that generator j will be dispatched only if the total net load is larger than the aggregated production of the generators that rank lower in terms of cost.

To encode this information, we construct a set of hyperplanes $\{\mathbf{1}^\intercal \mathbf{d} \geq \sum_{i=1}^j \overline{p}_i\}$ for $j \in \mathcal{G}$. That is, each hyperplane corresponds to a supply curve that renders the corresponding generator as the marginal generator, and checks whether the aggregated demand exceeds the total generation capacity.

2) Network congestion splits: Here, we propose hyperplane splits that encode information about expected network congestion conditioned on input net load profiles. To this end, we train a set of classifiers, namely SVMs [25] with linear kernel to predict whether a line gets congested. However, we do not use the SVMs for out-of-sample prediction; instead, we retrieve the maximum margin hyperplane learned for each SVM and use it as a candidate split in the tree learning process.

The process is described as follows. First, we inspect the full training data set $\mathcal{D}$ for line congestions. For each congested line, we formulate a binary classification problem with the line status as the target variable and the net load observations $\mathbf{d}_i$ as features. We then train an SVM model with linear kernel for each classification task. The SVM models effectively learn a separating hyperplane, parameterized by linear coefficients $\mathbf{w}$ and the intercept b. These seperating hyperplanes are subsequently used as candidate splits during the decision tree

[1]Note that globally optimal trees [9] address this shortcoming at the expense, however, of increased computational cost.

learning process, as described in Algorithm (1).

C. Dealing with Large-scale Problems

Training the proposed affine prescriptive trees requires solving (5) repeatedly during training. Specifically, for a tree of depth δ, assuming K candidate splits at root node, problem (5) need to be solved up to $\sum_{\delta=0}^{\delta^{max}} = 2^{\delta}(K-\delta)$ times during the offline training phase. However, the training process might become computationally prohibitive for larger networks. To mitigate this issue, we explore two directions to reduce the offline computational cost, namely, to speed up the solution of (5) and to reduce the time to find the polyhedral partition.

Firstly, we use an iterative algorithm to speed up the solution of (5). Section II-C uses duality theory to reformulate (5) as a deterministic optimization problem. Depending on the problem instance, however, iterative cutting-plane methods may be faster [26]. Here, we propose an intermediate approach that leverages the fact that only a small number of line constraints are binding. We initialize our master problem by reformulating (5b) and (5d) using duality, ignoring all line constraints (5c). Next, we solve the master problem and retrieve $\mathbf{W}_\ell^*, \mathbf{b}_\ell^*$. We then iterate over all the lines, fix the affine decision rules, and estimate the worst-case constraint violation, which is a maximization problem over $\mathbf{d}$. The line that leads to the highest violation is selected, and the respective row of (5c) is reformulated via duality and added to the master problem. The algorithm terminates when there is no violation. The training data set $\mathcal{D}$ can also inform us of which lines might lead to violations; thus, we can warm-start the iterative algorithm by adding these lines in the initial master problem. In this case, the algorithm typically terminates after a small number of iterations.

Secondly, we propose a surrogate tree-learning algorithm that "relaxes" the training process, thus reducing the time to find the polyhedral partition. Instead of training the tree in a fully prescriptive fashion as detailed in Algorithm 1, we take a sequential approach. Fist, we grow a decision tree minimizing the MSE (6) criterion with no constraints, for which a closed-form solution exists, and maintain the semi-greedy split selection. After retrieving an N_{min}-admissible polyhedral partition, we iterate over each leaf and estimate the affine decision rules that minimize the within-leaf dispatch cost by solving (5). Note that the original algorithm jointly estimates the polyhedral partition and the policy, i.e., the affine decision rules, in a semi-greedy, top-down fashion. The "relaxed" version, on the other hand, takes a sequential approach: first, we find the polyhedral partition, then we learn the affine decision rules.

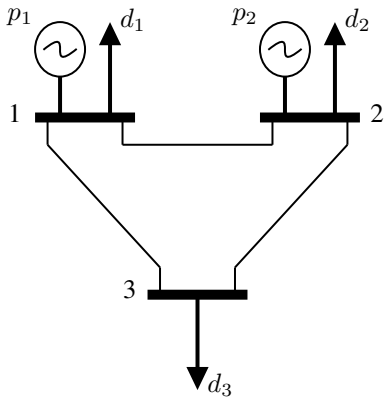
These approaches significantly reduce the offline computational cost, making the proposed tree-based method computationally tractable for larger networks.

IV. Illustrative Example

We illustrate the most salient features of our approach using the 3-bus system from the PGLib-OPF library [27], which we modify by setting the maximum capacity of the cheapest generator at $\overline{p}_2 = 270$MW, and the capacity of the line connecting buses 2 and 3 at $\overline{f}_2 = 90$MW —see Fig. 1 for details. If neither of these limits are reached, then at the optimal solution $p_1^* = 0$ and $p_2^* = \mathbf{1}^\intercal\mathbf{d}$; else, $p_1^* > 0$. We further assume that $d_1 = 110\,\text{MW}$ and that d_2, d_3 follow a multivariate normal distribution $N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, where

$$\boldsymbol{\mu} = (99, 81)\text{MW}, \; \boldsymbol{\Sigma} = \begin{bmatrix} 30.25 & 15.75 \\ 15.75 & 22.65 \end{bmatrix} \text{MW}^2,$$

are the mean vector and covariance matrix, respectively, and lie within intervals $d_2 \in [88, 110]$ MW and $d_3 \in [57, 95]$ MW.



Gen.	Cost ($/MWh)	$\overline{p}$ (MW)	Line	From Bus	To Bus	X	$\overline{f}$ (MW)
p_1	5	1000	1	1	3	0.62	9000
p_2	1.2	270	2	3	2	0.75	90
			3	1	2	0.9	9000

Fig. 1. Modified 3-bus system.

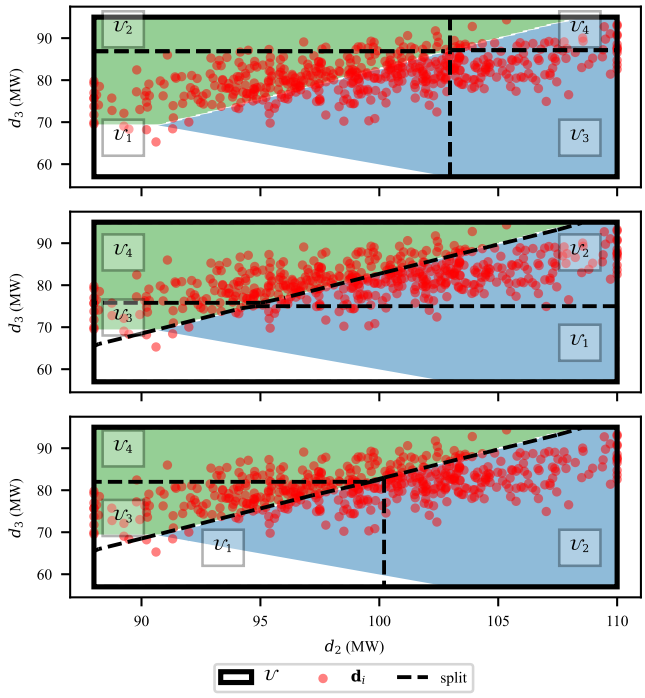


Fig. 2. Top: Tree with parallel splits. Middle: Tree with hyperplane splits. Bottom: Tree with hyperplane splits, trained with the surrogate method. Colored subregions indicate critical regions, red points indicate training observations. Solid lines show the load domain, $\mathcal{U}_i$ represents the i-th leaf.

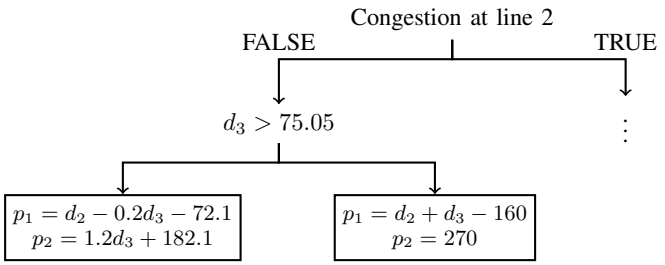


Fig. 3. Visualization of piecewise affine policy.

We generate $1\,000$ random observations and apply a $50/50$ training/test split to examine the performance of prescriptive trees with respect to (*w.r.t.*) hyperplane splits, setting $\delta^{max} = 2$ and $N_{min} = 25$. Performance is evaluated by estimating the mean increase in decision cost over the test set compared to a traditional LP solver. Three models are trained: one using only parallel splits, one using both parallel and hyperplane splits, and one using both splits but trained with the surrogate method developed in Section III-C. For parallel splits, we examine 9 equally spaced quantiles estimated from the training observations. For hyperplane splits, we consider a merit order split that checks whether $\mathbf{1}^\intercal\mathbf{d} \geq \overline{p}_2$ and a network congestion split derived from an SVM that predicts when line 2 gets congested.

Fig. 2 plots the tree splits as a function of d_2, d_3, where the colored subregions indicate the load profiles for which the set of active constraints does not change. Specifically, the green subregion indicates line 2 is congested, the blue subregion indicates that the maximum capacity of the cheapest generator is reached ($p_2^* = \overline{p}_2$), and the white subregion indicates that no upper limit is reached ($p_1^* = 0$, $p_2^* = \mathbf{1}^\intercal\mathbf{d}$). Evidently, the optimal policy is piecewise linear *w.r.t.* each subregion, and a tree that recovers this partition would yield an optimal policy.

Considering only parallel splits cannot recover a near optimal partition and leads to an out-of-sample mean cost increase of 3.19%—see top of Fig. 2. Conversely, hyperplane splits lead to significantly better decisions with an out-of-sample mean cost increase of 0.37%, as the root node is split at the hyperplane provided by the SVM — see middle of Fig. 2. A small decision error persists as the critical regions are not recovered exactly by the polyhedral partition; thus, leaves that extend to more than one subregions, i.e., $\mathcal{U}_3, \mathcal{U}_1$, lead to slightly suboptimal decisions. The surrogate method with hyperplanes leads to a mean cost increase of 1.72%, which ranks in-between the other models. Compared to the fully prescriptive method, the increased cost of the surrogate algorithm is attributed to the selection of parallel splits. First, the parallel split on $d_2 > 100.36$ creates two partitions that extend over two critical regions—see $\mathcal{U}_1, \mathcal{U}_2$ in the bottom of Fig. 2. Second, the split that separates $\mathcal{U}_3, \mathcal{U}_4$ ($d_3 > 81.78$) leads to a similar number of observations at each leaf. Conversely, the respective split at the middle of Fig. 2 ($d_3 > 75.75$) explicitly maximizes the coverage of each critical region, i.e., maximizes the area of $\mathcal{U}_4$. Interestingly, the merit order split is not selected in neither case. Even though it perfectly separates the white from the blue subregion, there are too few observations within the white region to merit splitting a node there. If d_2 and d_3 were, in contrast, uniformly distributed within their respective intervals, the merit order split would become highly prescriptive and, thus, selected by Algorithm 1.

Fig. 3 provides an interpretable visualization of the piecewise affine policy of the prescriptive tree with hyperplanes (middle of Fig. 2), focusing on $\mathcal{U}_1, \mathcal{U}_2$. Intuitively, the root node examines if a congestion in line 2 is expected; if not, then we evaluate d_3. If $d_3 > 75.05$, i.e., we reach $\mathcal{U}_2$, then $p_2 = \overline{p}_2$ and p_1 covers the excess demand (recall that $d_1 = 110\text{MW}$). Conversely, when $d_3 <= 75.05$, we reach $\mathcal{U}_1$, which extends to the white and blue subregions; here, both p_1, p_2 linearly depend on the varying demands.

V. Numerical Experiments

In this section, we describe our experimental setup (Subsection V-A), we present our main results (Subsection V-B), and provide additional results under challenging operating conditions (Subsection V-C). The code to reproduce the results is made available in [28].

A. Experimental Setup

1) Test Cases: The proposed methodology is demonstrated on a range of PGLib-OPF networks v21.07 [27] of up to 300 buses.

2) Load Data: The net load domain is defined as $\mathcal{U} = \{\overline{\mathbf{d}} - 0.4|\overline{\mathbf{d}}| \leq \mathbf{d} \leq \overline{\mathbf{d}}\}$, where $\overline{\mathbf{d}}$ denotes the nominal load values from the base case specified in [27]. Thus, positive loads vary within 60% and 100% of their nominal value. Two settings *w.r.t.* uncertainty are considered. First, each net load is independently and uniformly distributed within $\mathcal{U}$. Second, the net loads follow a multivariate normal distribution. For each net load d_j, the mean value is set at $\mu_j = 0.8\overline{d}_j$ and its standard deviation at $\sigma_j = 0.05\overline{d}_j$. We further sample correlations across net loads uniformly from $[0, 1]$ and use it to create the covariance matrix. In both cases, we generate $20\,000$ samples, and apply a $50/50$ training/test split.

3) Benchmarks: The following models are examined:

- `APT`: an affine prescriptive tree using only parallel splits.
- `APTH`: an affine prescriptive tree using both parallel and hyperplane splits.
- `APTH-rlx`: an affine prescriptive tree using both parallel and hyperplane splits and trained with the surrogate algorithm of Section III-C.
- `NN-prj`: an NN-based end-to-end learning model, coupled with an additional projection step.

For the tree-based models, namely `APT`, `APTH`, `APTH-rlx`, we set $N_{min} = 25$ and $\delta^{max} = 3$, which are values that enable interpretability and avoid overfitting. Parallel splits are evaluated at 19 equally spaced quantiles estimated from the training observations. We further consider a hard time-limit constraint of $10\,000$ seconds; that is, if the time limit is reached, we stop growing the tree and each node becomes a leaf. For the larger networks, i.e., case118, case300, we use the iterative algorithm described in Section III-C to solve (5).

TABLE I
NUMBER OF CONGESTED LINES, NUMBER OF SVM MODELS TRAINED, AND CLASSIFIER ACCURACY (%).

	Uniform		Normal	
	No. lines/models	mean/min acc. (%)	No. lines/models	mean/min acc. (%)
case5	1/1	99.97	1/1	99.99
case30	1/1	99.79	1/1	99.91
case39	2/1	99.83	2/1	99.60
case57	0/0	-	0/0	-
case118	5/3	93.10/80.96	5/4	95.72/84.60
case300	13/8	97.59/93.36	17/8	96.83/89.92

TABLE II
PERCENTAGE (%) OF MCI, $\delta^{max} = 3$. PARENTHESES SHOW THE RATE OF INFEASIBILITY (%).

	Uniform				Normal			
	APT	APTH	APTH-rlx	NN-prj	APT	APTH	APTH-rlx	NN-prj
case5	1.62	**0.40**	0.46	0.96 (5.35)	**0.30**	0.33	1.39	0.86 (1.61)
case30	4.20	0.76	0.85	**0.52 (5.12)**	1.88	1.19	1.58	**0.60 (14.18)**
case39	2.07	0.22	0.23	**0.21 (3.37)**	1.54	**0.16**	0.48	0.35 (1.17)
case57	**0.00**	**0.00**	**0.00**	0.18 (0.11)	**0.00**	**0.00**	**0.00**	0.18 (0.27)
case118	1.17	0.42	0.28	**0.19 (7.73)**	1.17	1.14	0.37	**0.16 (21.85)**
case300	3.10	2.81	2.44	**1.80 (43.29)**	3.12	3.12	2.43	**1.20 (59.48)**

For NN-prj, we consider a multi-layer feed-forward structure with 4 hidden layers and 100 nodes per layer, using the MSE loss and the ReLU activation function in the hidden layers. Following [4], [5], [8], we apply a sigmoid activation function in the output layer, thus ensuring that the predicted decisions satisfy the generation capacity constraints. We further add a regularization term in the objective that penalizes excessive line flows, following [5]. The rest of the hyperparameters are also set according to [5] and the NN model is trained with early stopping to avoid overfitting. An ℓ_1-projection step is applied post-hoc to ensure feasible decisions. For the ground truth solution of the DC-OPF problem, we use the Gurobi solver [29] with default settings. All experiments are run on a standard PC featuring Intel Core i7 CPU with a clock rate of 2.7 GHz and 16GB of RAM.

4) Perfomance Metrics: For performance evaluation, we measure the suboptimality of predicted decisions by estimating the percentage of mean cost increase (MCI) over a test set of N_{test} observations, given by

$$\frac{1}{N_{\text{test}}} \sum_{i \in [N_{\text{test}}]} \frac{\mathbf{c}^\intercal(\hat{\mathbf{p}}_i - \mathbf{p}_i^*)}{\mathbf{c}^\intercal \mathbf{p}_i^*},$$

where $\mathbf{p}_i^*$ is the optimal solution derived from Gurobi and $\hat{\mathbf{p}}_i$ the predicted solution for the i-th test sample. Evidently, MCI is non-negative.

B. Results

Table I summarizes the results of the SVM classifiers, namely the number of lines that face congestion at least once, the number of SVM classifiers trained, and the average and minimum classifier accuracy (%) per test case. Note that to train an SVM classifier we require at least N_{min} observations per class label; that is, if a line is almost always or almost never congested, we do not train a model— see, e.g., case39, case118, and case300. For the small and medium-sized cases, the SVM models provide almost perfect separation with close to 100% accuracy. For the larger cases, i.e., case118 and case300, the average accuracy still exceeds 93% for both uniform and normal distribution. However, there is an increased variability in individual models, as indicated by the worst-case performance. This is more pronounced for case118, where the worst-case performance is below 85% for both types of uncertainty distribution.

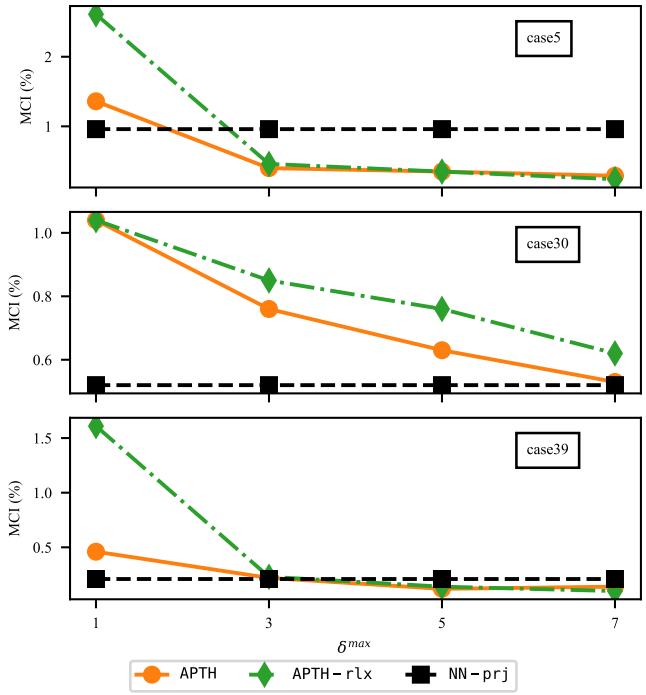


Fig. 4. MCI versus maximum tree depth δ^{max} (uniform uncertainty).

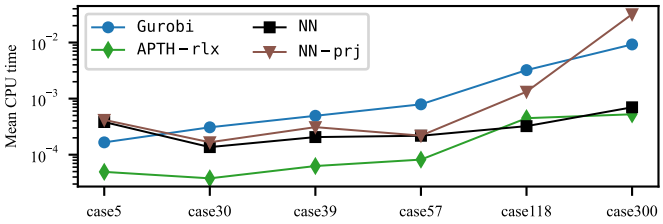


Fig. 5. Mean CPU time to solve a single problem instance. NN denotes the inference time of the NN-based model without projection. The y-axis is in logarithmic scale.

Table II presents the out-of-sample MCI for the examined test cases. For NN-prj, we also report the percentage of infeasible solutions. Clearly, the tree-based solutions are always feasible by design. In almost all cases, the lowest MCI is smaller than 1%, which is on par with previous works. The worst performance is observed for case300, which is probably attributed to the large number of lines facing congestion.

Overall, considering hyperplane splits significantly improves the prescriptive performance of the tree-based method. Specifically, the average (maximum) improvement of APTH compared to APT is 53% (89%) for uniform distribution and 20% (90%) for normal distribution, respectively. The only exception is for case5 and normal distribution, where APT is 10% better than APTH. Evidently, the effect of hyperplanes

is more pronounced when net loads are uniformly distributed, as we observe that APT performs, on average, much better under a normal distribution. This could be attributed to the training data extending to a smaller number of critical regions when loads are normally distributed, which, in turn, nullifies the impact of a number of candidate hyperplanes.

We further observe that prescriptive trees perform competitively with NN−prj in terms of decision performance, resulting in a lower MCI in $5/12$ cases examined. However, a significant percentage of NN−prj solutions may be infeasible and require a projection step. The rate of infeasibility seems to be increasing with the size of the network, with the worst-case being observed for case118 and case300, for both types of uncertainty.

Now we discuss the efficacy of the surrogate learning algorithm proposed in Section III-C. When APTH is fully grown, i.e., the algorithm terminates before the imposed time limit is reached, it outperforms APTH−rlx, with the differences being small in general, except for case57, where both are optimal. For case188 and case300, the time limit is reached before APTH is fully grown, which leads to APTH−rlx outperforming APTH. Moreover, APTH−rlx outperforms APT, which does not consider hyperplane splits, in all cases but one, and is on par with NN−prj. Notably, APTH−rlx reduces the training time by over 95% in all cases compared to APTH; thus, APTH−rlx achieves a good trade-off between computational efficiency and prescriptive performance.

The results presented in Table II concern shallow trees ($\delta^{max} = 3$). Evidently, increasing the tree depth is expected to improve decision performance. We investigate this claim by evaluating the sensitivity of decision quality *w.r.t.* the maximum tree depth δ^{max}. Fig. 4 plots the out-of-sample MCI of APTH and APTH−rlx as a function of δ^{max} for three test cases and uniform uncertainty; the performance of NN−prj is also plotted for reference. In all examined cases, increasing δ^{max} leads to significant gains in performance for APTH and APTH−rlx, with the relative improvement being more pronounced for smaller values of δ^{max}. Moreover, APTH converges to better performance than NN−prj as δ^{max} increases, with a relatively small depth of $\delta^{max} = 5$ being sufficient for adequate performance.

We further investigate whether end-to-end learning improves over Gurobi in terms of inference speed. Fig. 5 plots the mean CPU time to solve or predict a single problem instance for a selection of models for uniform uncertainty (y-axis is in logarithmic scale). We denote NN as the NN-based model prior to projection. For NN−prj, we sum the inference time of NN and the time to solve the projection step, weighted by the probability of infeasibility. For Gurobi, we only consider CPU time to solve the problem and not the time to formulate it. As all tree-based models exhibit similar inference time, we only plot APTH−rlx.

Overall, APTH−rlx consistently leads to smaller CPU time compared to both Gurobi and NN−prj, and even outperforms NN. As expected, the mean CPU time of Gurobi increases with the size of the network. The NN−prj performance varies with its out-of-sample infeasibility rate. For medium to large-sized cases, when the infeasibility rate is below 10%, e.g., case30 through case118, the inference time of NN−prj is smaller than that of Gurobi. However, for case300, when the infeasibility rate reaches over 40%, the required projection step negates any improvement in inference speed and nullifies the intended purpose of applying end-to-end learning in the first place.

TABLE III
NUMBER OF CONGESTED LINES, NUMBER OF SVM MODELS TRAINED, AND CLASSIFIER ACCURACY (%), API TEST CASES.

	Uniform		Normal	
	No. lines/models	mean/min acc. (%)	No. lines/models	mean/min acc. (%)
case5_api	3/3	99.83/99.59	2/1	99.95
case30_api	0/0	-	0/0	-
case39_api	10/4	97.59/92.92	7/4	99.46/98.81
case57_api	0/0	-	0/0	-
case118_api	16/11	95.52/85.54	15/12	94.56/78.81
case300_api	16/10	94.47/72.07	14/13	95.11/65.66

TABLE IV
PERCENTAGE (%) OF MCI, $\delta^{max} = 3$, API TEST CASES. PARENTHESES SHOW THE RATE OF INFEASIBILITY (%).

	Uniform				Normal			
	APT	APTH	APTH−rlx	NN−prj	APT	APTH	APTH−rlx	NN−prj
case5_api	0.05	**0.01**	0.02	0.85 (0.68)	0.02	**0.01**	0.13	0.71 (0.71)
case30_api	**0.00**	**0.00**	**0.00**	0.73 (3.03)	**0.00**	**0.00**	**0.00**	0.49 (3.01)
case39_api	0.93	0.65	0.75	**0.47 (13.11)**	0.57	0.52	**0.40**	0.68 (17.19)
case57_api	0.49	**0.00**	**0.00**	0.05 (0.03)	0.34	**0.00**	**0.00**	0.03 (0.02)
case118_api	22.07	17.65	16.04	**3.19 (86.91)**	19.27	18.74	18.36	**4.03 (93.08)**
case300_api	3.36	2.68	1.87	**1.52 (71.95)**	3.43	2.67	2.08	**1.75 (74.15)**

C. Results for More Challenging Test Cases

To evaluate the sensitivity *w.r.t.* number of lines that face congestion, we repeat the previous experiment on more challenging test cases. Specifically, we examine performance on the *active power increase* (api) test cases [27], where the nominal $\mathbf{d}$ is increased.

Table III presents the performance of the SVM classifiers on the more challenging test cases. Compared to Table I, it is evident that the api test cases face congestion more frequently. For the smaller cases, the SVMs still perform well, with an average accuracy of over 97%. For case188_api and case300_api, the average accuracy remains around 95% for both types of uncertainty. However, we observe large variability based on the worst-case SVM performance, which is more pronounced for case300_api, where the worst-case performance is approximately 72% and 66% for uniform and normal distributions, respectively.

Table IV presents the out-of-sample MCI under the more challenging operating conditions, alongside the infeasibility rate for NN−prj. Compared to Table II, we observe an increase in MCI for larger networks, which is attributed to the more challenging nature of the underlying problems. This is especially pronounced for case118_api where the number of lines that face congestion is three times larger than case118. Furthermore, the infeasibility rate of NN−prj increases significantly for the larger cases, with an average infeasibility rate of approximately 90% for case118_api and 74% for case300_api, indicating the difficulty in predicting feasible decisions.

In terms of relative performance, results are consistent with the previous experiments. Specifically, APTH consistently outperforms APT, while APTH−rlx performs similarly to

APTH and outperforms APT. Interestingly, APTH-rlx even outperforms APTH for case39_api and normally distributed net loads. When comparing the tree-based models with NN-prj, we observe that NN-prj perform better only when their infeasibility rate is high. Notably, NN-prj leads to significantly lower MCI for case118_api and case300_api for both types of uncertainty, but has a high infeasibility rate in both cases. However, as previously shown in Fig. 5, a high infeasibility rate negates the respective gains of end-to-end learning over the traditional LP solver in terms of inference speed, making the choice of NN-prj counterproductive.

VI. Conclusions

This work presented an interpretable approach for end-to-end learning of DC-OPF solutions with feasibility guarantees. We developed prescriptive decision trees that learn a piecewise affine mapping from varying load data to DC-OPF solutions, using robust optimization to ensure feasibility of decisions. We proposed domain-informed hyperplane splits, modeling the merit order curve and network congestion; for the latter, we utilized SVM classifiers that model expected line congestion as a function of varying load data. A comprehensive evaluation was conducted considering a number of test cases, different types of uncertainty, and various operating conditions. The results highlighted the efficacy of the proposed domain-informed hyperplane splits, which led to an average performance increase of 36% compared to tree models using only parallel splits. Further, shallow prescriptive trees with hyperplanes of maximum depth of 3 outperformed NN-based benchmarks in approximately 46% of the experiments; a sensitivity analysis *w.r.t.* model complexity illustrated that the performance of prescriptive trees further improved as their depth increased. The proposed approach was also significantly faster than a state-of-the-art LP solver. Additional experiments under challenging operating conditions further validated the efficacy of the proposed approach. Overall, this study highlighted the benefits of encoding domain knowledge during model development, which not only achieves comparable performance to black box, state-of-the-art benchmarks but also enables interpretability.

Future work may explore mapping contextual information, e.g., calendar variables or temperature forecasts, to OPF decisions. Another direction to explore is to use non-linear functions to separate classes that also retain the computational tractability of the proposed policy, e.g., SVMs with piecewise linear feature mapping.

References

[1] B. Stott, J. Jardim, and O. Alsac, "Dc power flow revisited," *IEEE Transactions on Power Systems*, vol. 24, no. 3, pp. 1290–1300, 2009.

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

[3] S. Chatzivasileiadis, A. Venzke, J. Stiasny, and G. Misyris, "Machine learning in power systems: Is it time to trust it?" *IEEE Power and Energy Magazine*, vol. 20, no. 3, pp. 32–41, 2022.

[4] X. Pan, T. Zhao, and M. Chen, "DeepOPF: Deep neural network for DC optimal power flow," in *2019 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)*, 2019, pp. 1–6.

[5] X. Pan, T. Zhao, M. Chen, and S. Zhang, "DeepOPF: A deep neural network approach for security-constrained DC optimal power flow," *IEEE Transactions on Power Systems*, vol. 36, no. 3, pp. 1725–1735, 2021.

[6] A. Venzke, G. Qu, S. Low, and S. Chatzivasileiadis, "Learning optimal power flow: Worst-case guarantees for neural networks," in *2020 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)*. IEEE, 2020, pp. 1–7.

[7] R. Nellikkath and S. Chatzivasileiadis, "Physics-informed neural networks for minimising worst-case violations in DC optimal power flow," in *2021 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)*, 2021, pp. 419–424.

[8] T. Zhao, X. Pan, M. Chen, and S. H. Low, "Ensuring DNN solution feasibility for optimization problems with convex constraints and its application to DC optimal power flow problems," *arXiv:2112.08091*, 2021.

[9] J. W. Dunn, "Optimal trees for prediction and prescription," Ph.D. dissertation, Massachusetts Institute of Technology, 2018.

[10] L. Wehenkel, T. Van Cutsem, and M. Ribbens-Pavella, "An artificial intelligence framework for online transient stability assessment of power systems," *IEEE Transactions on Power Systems*, vol. 4, no. 2, pp. 789–800, 1989.

[11] J. L. Cremer, I. Konstantelos, and G. Strbac, "From optimization-based machine learning to interpretable security rules for operation," *IEEE Transactions on Power Systems*, vol. 34, no. 5, pp. 3826–3836, 2019.

[12] Y. Ng, S. Misra, L. A. Roald, and S. Backhaus, "Statistical learning for DC optimal power flow," in *2018 Power Systems Computation Conference (PSCC)*, 2018, pp. 1–7.

[13] S. Misra, L. Roald, and Y. Ng, "Learning for constrained optimization: Identifying optimal active constraint sets," *INFORMS Journal on Computing*, vol. 34, no. 1, p. 463–480, 2022.

[14] D. Deka and S. Misra, "Learning for DC-OPF: Classifying active sets using neural nets," in *2019 IEEE Milan PowerTech*, 2019, pp. 1–6.

[15] Y. Chen and B. Zhang, "Learning to solve network flow problems via neural decoding," *arXiv:2002.04091*, 2020.

[16] A. Robson, M. Jamei, C. Ududec, and L. Mones, "Learning an optimally reduced formulation of OPF through meta-optimization," *arXiv:1911.06784*, 2019.

[17] L. Chen, M. Sim, X. Zhang, and M. Zhou, "Robust explainable prescriptive analytics," *Available at SSRN 4106222*, 2022.

[18] I. Pappas, D. Kenefake, B. Burnak, S. Avraamidou, H. S. Ganesh, J. Katz, N. A. Diangelakis, and E. N. Pistikopoulos, "Multiparametric programming in process systems engineering: Recent developments and path forward," *Frontiers in Chemical Engineering*, vol. 2, p. 620168, 2021.

[19] F. Zhou, J. Anderson, and S. H. Low, "The optimal power flow operator: Theory and computation," *IEEE Transactions on Control of Network Systems*, vol. 8, no. 2, pp. 1010–1022, 2020.

[20] E. T. Maddalena, R. K. H. Galvão, and R. J. M. Afonso, "Robust region elimination for piecewise affine control laws," *Automatica*, vol. 99, pp. 333–337, 2019.

[21] D. Bertsimas and D. den Hertog, *Robust and adaptive optimization*. Dynamic Ideas LLC, 2020, vol. 958.

[22] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen, *Classification and regression trees*. CRC press, 1984.

[23] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[24] S. K. Murthy, S. Kasif, S. Salzberg, and R. Beigel, "Oc1: A randomized algorithm for building oblique decision trees," in *Proceedings of AAAI*, vol. 93. Citeseer, 1993, pp. 322–327.

[25] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, pp. 273–297, 1995.

[26] D. Bertsimas, I. Dunning, and M. Lubin, "Reformulation versus cutting-planes for robust optimization: A computational study," *Computational Management Science*, vol. 13, pp. 195–217, 2016.

[27] S. Babaeinejadsarookolaee, A. Birchfield, R. D. Christie, C. Coffrin, C. DeMarco, R. Diao, M. Ferris, S. Fliscounakis, S. Greene, R. Huang *et al.*, "The power grid library for benchmarking ac optimal power flow algorithms," *arXiv:1908.02788*, 2019.

[28] https://git.persee.mines-paristech.fr/akylas.stratigakos/interpretable-learning-dcopf.

[29] Gurobi Optimization, LLC, "Gurobi Optimizer Reference Manual," 2023.