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, *Member, IEEE,* Salvador Pineda, *Member, IEEE,* Juan Miguel Morales, *Senior Member, IEEE,* and Georges Kariniotakis, *Senior Member, IEEE*

***Abstract*—The increased uncertainty due to the integration of stochastic renewable energy sources necessitates solving Optimal Power Flow (OPF) problems repeatedly and with greater granularity. 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 non-orthogonal splits informed by domain knowledge, including network congestion and the merit order curve. By incorporating this information, we 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.**

***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 its original form, the OPF problem is a non-convex problem that is difficult to solve. In various important use cases, a linearized version of the OPF that considers only active power, referred to as DC-OPF [1], is utilized. In particular, the DC-OPF is the cornerstone of deregulated electricity markets as it is widely adopted to determine locational marginal prices which are influenced by network congestion. The DC-OPF is also important for risk assessment and ensuring a reliable operation by considering variants that incorporate steady-state security constraints, such as the Security Constrained DC-OPF. The DC-OPF problem is especially appealing as it can be expressed as a linear programming (LP) problem that can be solved efficiently.

Although general-purpose optimization solvers have made solving LP problems efficient, certain settings can present computational challenges. To cope with the increased uncertainty and variability due to the integration of renewable production, future electricity markets are expected to move closer to real-time, e.g., operating on a 5-minute ahead basis [2]. In turn, this necessitates solving DC-OPF problems repeatedly and at a higher speed and scale, while respecting stringent time constraints, e.g., in less than 1-minute [3, Exhibit 2.1]. Even though state-of-the-art LP solvers are adequate most of the time, they have a worst-case complexity that scales polynomially with the size of the grid, which may create a computational bottleneck in some instances. Hence, a fallback solution is needed.

Machine learning (ML) has been rapidly evolving in recent years, revolutionizing many industries, including power systems [4]. ML-based methods can complement traditional optimization solvers in various problems, such as the DC-OPF, by shifting the computational cost from inference to the offline training phase. For instance, system operators, like Midcontinent Independent System Operator (MISO), perform a scenario-based, real-time risk assessment considering a 5-minute window, which corresponds to solving 288 LP problems. A state-of-the-art LP solver requires approximately 15 minutes to evaluate a single scenario, while an ML-based proxy solver takes less than 5 seconds [5], thus allowing for the evaluation of a much larger number of scenarios and, ultimately, enabling a more reliable system operation. However, power systems are critical infrastructure and stakeholders are naturally risk averse, which presents obstacles to the adoption of these tools [6]. Transparency, interpretability, and performance guarantees are necessary for the practical implementation of ML models for problems such as the DC-

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.

OPF. For instance, European Union legislation establishes the need for the so-called "right to explanation" [7], i.e., the requirement of automated systems to provide information about their internal logic, which necessitates interpretable and transparent methods. 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 learning an optimization proxy. 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 [5], [8]–[12]. For instance, [8] proposes an NN model with a constraint violation penalty to predict the DC-OPF solutions; a similar model is developed in [9] for Security Constrained DC-OPF. To ensure the feasibility of decisions, both models require a post hoc projection step. This projection onto the feasible set is itself an optimization problem that needs to be solved, which might be of the same complexity as the original problem, and may potentially negate any computational benefits. In [10], 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 [11], physics-informed NNs demonstrate improved guarantees over standard NNs. However, ensuring that predicted decisions satisfy the problem constraints remains a challenge for end-to-end learning methods as prediction errors are inevitable. To address this issue, [12] 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. Additionally, [5] embeds a closed-form, repair layer within an NN model, which ensures the feasibility of a subset of problem constraints. Overall, NN-based models have the modeling capacity to approximate the optimal function that maps load profiles to problem decisions. Nevertheless, even with feasible solutions guaranteed, NN models still lack the interpretability of other ML methods, such as decision trees [13], which is critical for adoption in real-world applications, especially in critical infrastructure. Decision trees are inherently interpretable and have been used in power systems for decades — see, e.g., [14] for an early and [15] 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 [16]–[20]. Specifically, [16] and [17] 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 [18], the problem of finding the active sets of constraints is formulated as a multiclass classification task. A neural decoding strategy is developed in [19] 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, [20] proposes a two-step process that combines the prediction of an 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 methods. 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. For instance, [17] finds that the number of critical regions observed for various networks is less than 10 and that this number is not correlated with the network size but rather depends on the load distribution and other network characteristics.

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 [21], 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 [22] 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 novel, two-step learning algorithm that combines axis-parallel and *domain-informed*, non-orthogonal splits that encode network information, namely the merit order curve and network congestion. We formulate the expectation of network congestion, conditioned on load profiles, 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 the feasibility of the predicted decisions for the whole parameter space in a principled manner. A surrogate learning algorithm is further 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 the feasibility of decisions. To the best of our knowledge, our work is the first to develop interpretable 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 (Section II-A), describes the proposed learning problem (Section II-B), and illustrates how to reformulate it into a tractable problem (Section 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 Power Transfer Distribution Factors (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 (Bounded uncertainty): 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 [23].

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.

Instead of solving (1), our goal is to learn a function (policy) that maps realizations of net load injections $\mathbf{d}$ to generator setpoints $\mathbf{p}$. From the theory of multiparametric programming [22], we know that the optimal dispatch $\mathbf{p}^*$ with respect to (*w.r.t.*) $\mathbf{d}$ takes the form of a piecewise affine function defined over a polyhedral partition of the feasible space. First, we define a polyhedral partition of the feasible space $\mathcal{U}$.

Definition 1 (Polyhedral partition [24]): 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.

If the polyhedral partition $\{\mathcal{U}_\ell\}_{\ell=1}^L$ recovers the critical regions of the parameter space, i.e., the regions where the active set of constraints at the optimal solution remains constant, then learning a piecewise affine function over $\{\mathcal{U}_\ell\}_{\ell=1}^L$ is optimal. An explicit solution for finding the optimal piecewise policy can be derived by recasting the problem as a multiparametric LP problem, but it might be intractable as the number of critical regions grows exponentially with the number of problem constraints in the worst case. In practice, however, only a small number of critical regions are relevant [17].

Since it is established that a piecewise affine policy is optimal, in this work, we propose learning a simpler, *data-driven* piecewise affine policy, which retains good performance and interpretability. 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

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

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 [22]. As shown in previous works [17], the number of critical regions populated with data observations is small in practice. The tree-learning algorithm developed in the next section effectively learns a partition of the form of Definition 2 that is as close as possible to the critical regions of the parameter space with data observations. In that case, N_{min}, which is a user-defined hyperparameter, corresponds to the minimum number of observations per each tree leaf and controls the complexity of the learned policy.

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 , i.e., each row of $\mathbf{W}_\ell$ defines a vector of coefficients that maps $\mathbf{d}$ to a specific generator, $\mathbf{b}_\ell$ 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

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

$$\text{s.t.} \quad \mathbf{W}_\ell^\intercal \mathbf{1} = \mathbf{1}, \tag{5b}$$

$$\mathbf{1}^\intercal \mathbf{b}_\ell = 0, \tag{5c}$$

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

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

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. Constraints (5b) (i.e., the row-wise sum of $\mathbf{W}_\ell$) and (5c) ensure that total generation equals total demand for all values of $\mathbf{d}$, thus satisfying the balancing constraint (1b)[1]. The robust constraints (5d), (5e) further ensure that, for all realizations of the $\mathbf{d}$ within $\mathcal{U}_\ell$, decisions satisfy the inequality constraints (1c), (1d), respectively. Effectively, by solving problem (5) for each $\mathcal{U}_\ell$ we learn the parameters of the proposed data-driven policy, which is of the form of Definition 3. 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 $f(\mathbf{d}_0) = \mathbf{W}_\ell \mathbf{d}_0 + \mathbf{b}_\ell$.

Formally, given an N_{min}-admissible polyhedral partition $\{\mathcal{U}_\ell\}_{\ell=1}^L$ that covers the whole feasible space $\mathcal{U}$ and robust constraints (5b) – (5e) that ensure forecast decisions are feasible $\forall \mathbf{d} \in \mathcal{U}_\ell$, it follows that the forecast decisions will always satisfy the constraints of (1) . Thus, we obtain guarantees about the feasibility of decisions for any realization of $\mathbf{d} \in \mathcal{U}$.

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, and has the added benefit of not requiring a set of solved instances $\{\mathbf{p}_i^*\}_{i=1}^N$ for training. Alternatively, given $\{\mathbf{p}_i^*\}_{i=1}^N$, the mean squared error (MSE) between 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 our primary focus is to minimize the total dispatch cost of forecast decisions, rather than their predictive error, we focus primarily on the prescriptive cost (5a).

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 [25] to reformulate (5) into a deterministic LP problem.

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

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

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

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

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 (5e) is replaced by the following constraints

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

[1]This follows by noting that $\mathbf{1}^\intercal(\mathbf{W}_\ell \mathbf{d} + \mathbf{b}_\ell) - \mathbf{1}^\intercal \mathbf{d} = 0$, $\forall \mathbf{d} \in \mathcal{U}_\ell$, provided that (5b), (5c) hold.

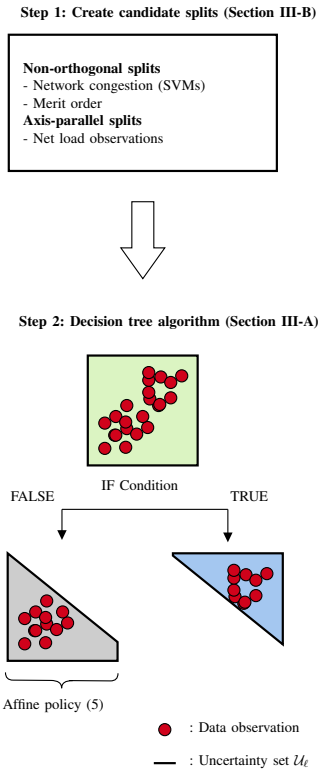


Fig. 1. Flowchart of the proposed two-step training process. Step 1 creates a set of candidate node splits. Step 2 uses the candidate node splits as input to grow a decision tree. The tree graph at the bottom visualizes a non-orthogonal node split.

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 (Section III-A). Next, we detail the process of finding domain-informed splits (Section III-B). Finally, we describe a surrogate learning method to deal with large problem instances (Section III-C). The two-step process to train the proposed tree-based model is illustrated in Fig. 1.

A. Affine Prescriptive Trees

In this section, we present our decision tree algorithm for learning a piecewise affine policy.

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 and, therefore, can be used to predict instances of the target variable given out-of-sample feature observations. Here, our primary goal is to use a tree-based algorithm to learn a polyhedral partition of the form of Definition 2 that is as close as possible to the critical regions of the parameter space, using data set $\mathcal{D}$.

Our proposed algorithm combines *axis-parallel* and *non-orthogonal* splits during the tree-learning process. To clarify, axis-parallel splits refer to splits that only consider a single feature, while non-orthogonal splits refer to splits that consider a linear combination of different features. Mathematically, both axis-parallel and non-orthogonal splits are represented as a set of hyperplanes. Using this combination of splits is a departure from most state-of-the-art tree algorithms that focus solely on binary trees with axis-parallel splits—see, e.g., [26] for single trees and [27] for tree-based ensembles. Oblique decision trees [28] allow for non-orthogonal splits and have been shown to lead to significant performance improvements; however, they can be computationally challenging and less interpretable [13]. To address this challenge, we construct a set of domain-informed non-orthogonal splits prior to the learning phase; the process of identifying these splits is detailed in Section III-B.

Algorithm 1 describes our decision tree algorithm in detail. Consider a root node (equivalently, partition) $\mathcal{U}_0 = \{\mathbf{d} \,|\, \mathbf{H}_0\mathbf{d} \leq \mathbf{h}_0\}$, a corresponding data set $\mathcal{D}_0$, and a set of K candidate hyperplanes to split on $\{(\boldsymbol{\alpha}_k, \beta_k)\}$, parameterized by vectors $\boldsymbol{\alpha}_k$ and scalars β_k. These hyperplanes model both non-orthogonal and axis-parallel splits as 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 $\mathcal{U}_l = \{\mathbf{d} \,|\, \boldsymbol{\alpha}_k^\intercal \mathbf{d} \leq \beta_k,\, \mathbf{d} \in \mathcal{U}_0\}$ 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 $\text{cl}(\mathcal{U}_r)$. Embedding (5) within the tree-learning algorithm ensures that node splits are selected based on their impact on the true decision cost of the DC-OPF problem. Specifically, 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. For reference, Fig. (1) visualizes splitting a tree node using a non-orthogonal split. 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 costs and map data to prescriptions. We take an intermediate approach to split selection, avoiding the well-known shortcoming of CART-like methods [26], which is determining each split without considering the possible impact on future splits[2]. Specifically, we apply a semi-greedy split selection, which prioritizes non-orthogonal splits over axis-parallel ones, as the former encode domain knowledge. To implement the semi-greedy split selection, we use an auxiliary

[2]Note that globally optimal trees [13] address this shortcoming using a mixed-integer LP formulation, at the expense, however, of increased computational cost.

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-orthogonal, we only evaluate the remaining non-orthogonal splits. If the current best split is axis-parallel, then we evaluate all the remaining splits, including the rest of the axis-parallel ones. This is described in Steps 4-5 of Algorithm 1, where $\neg, \wedge$ denote the logical negation and conjunction (*and*) operators, and the **continue** statement interrupts the current step of a loop and continues with the next iteration. This approach prioritizes domain-informed non-orthogonal splits while still allowing for data-driven axis-parallel splits to be considered if the former 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}, both controlling the complexity of the learned policy. Namely, N_{min} controls the bias-variance trade-off, with smaller values increasing the risk of overfitting, and 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 result in overfitting and reduced interpretability. The maximum number of partitions that can be recovered is $2^{\delta^{max}}$ and is independent of the size of the underlying network. For a sufficiently complex policy, i.e., one with small N_{min} and large δ^{max}, we expect that the number of partitions recovered scales with the number of critical regions that are populated with data observations. Thus, we avoid the shortcoming of multiparametric LP, where the number of partitions scales exponentially with the problem constraints. To promote interpretability and avoid potential overfitting, 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 $v_{min} \leftarrow |\mathcal{D}_0|\cdot v_0$, `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) $+ |\mathcal{D}_r| \cdot \min_{\mathbf{d}\in\mathrm{cl}(\mathcal{U}_r)}$ (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, Non-Orthogonal Splits

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

Axis-parallel splits only check whether an entry of $\mathbf{d}$ exceeds a threshold value; they are purely data-driven and the standard approach to growing binary trees, e.g., CART. In this work, the set of axis-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, non-orthogonal splits that are potentially more effective than data-driven axis-parallel splits. The proposed splits are derived from hyperplanes that 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 respective generator as the marginal one, and checks whether the aggregated demand exceeds the total generation capacity.

2) Network congestion splits: Here, we propose non-orthogonal 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 [29] with a 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 of creating non-orthogonal splits that model network congestion 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 a linear kernel for each classification task, which effectively learns a separating hyperplane, parameterized by linear coefficients $\mathbf{w}$ and the intercept b. These separating hyperplanes are subsequently used as candidate splits during the decision

tree learning process, as shown in Fig. 1 and detailed 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 the 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.

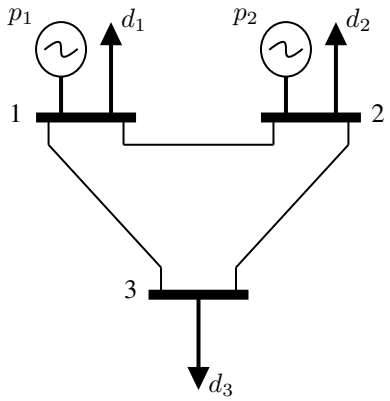
Firstly, we use an iterative algorithm to speed up the solution of (5). Section II-C uses duality theory to reformulate (5) as a deterministic optimization problem. Depending on the problem instance, however, iterative cutting-plane methods may be faster [30]. Here, we propose an intermediate approach that leverages the fact that only a small number of line constraints are binding. We initialize our master problem by reformulating (5e) using duality, ignoring all line constraints (5d). 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 (5d) 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 to 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. First, 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. The computational cost in this step is similar to training a standard CART-like tree. 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 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. The surrogate learning algorithm could also be utilized with more computationally demanding variants of the DC-OPF problem.

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 [31], which



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. 2. Modified 3-bus system.

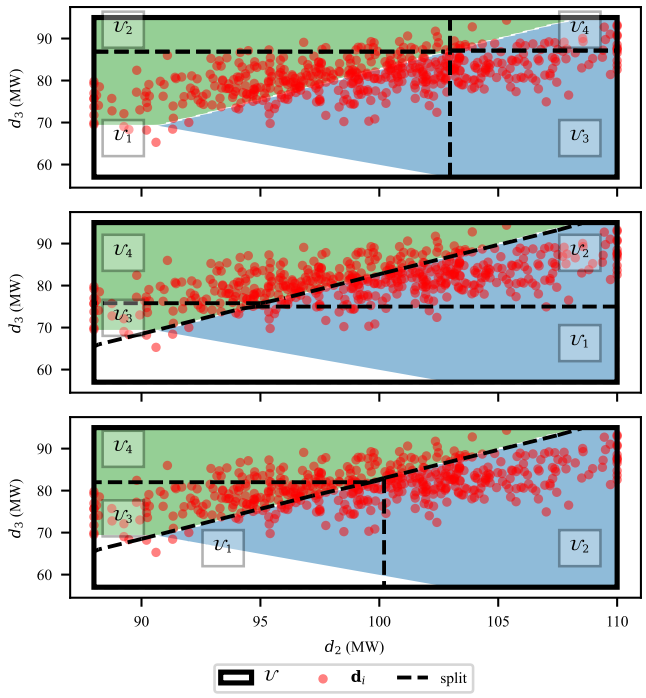


Fig. 3. Top: Tree with axis-parallel splits. Middle: Tree with non-orthogonal splits. Bottom: Tree with non-orthogonal splits, trained with the surrogate method. Colored subregions indicate critical regions and red points indicate training observations. Solid lines show the load domain, $\mathcal{U}_i$ represents the i-th leaf.

we modify by setting the maximum capacity of the cheapest generator at $\overline{p}_2 = 270\,\text{MW}$, and the capacity of the line connecting buses 2 and 3 at $\overline{f}_2 = 90\,\text{MW}$ — see Fig. 2 for details. If neither of these limits is 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

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

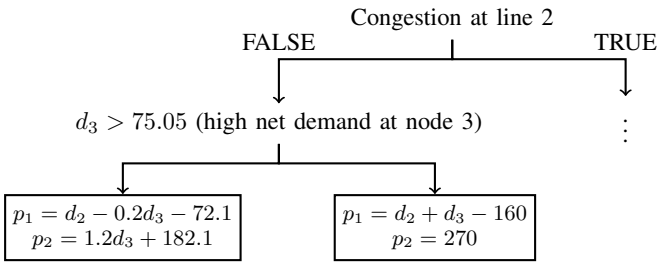


Fig. 4. Visualization of piecewise affine policy.

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.

We generate 1 000 random observations and apply a $50/50$ training/test split to examine the performance of prescriptive trees *w.r.t.* non-orthogonal 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 an LP solver. Three models are trained: one using only axis-parallel splits, one using both axis-parallel and non-orthogonal splits, and one using both types of splits but trained with the surrogate method developed in Section III-C. For axis-parallel splits, we examine 9 equally spaced quantiles estimated from the training observations. For non-orthogonal 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. 3 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 axis-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. 3. Conversely, non-orthogonal 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 the middle of Fig. 3. 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 subregion, i.e., $\mathcal{U}_3, \mathcal{U}_1$, lead to slightly suboptimal decisions. Specifically, perfectly separating between instances of line congestion (green subregion) and the rest requires a piecewise affine function. The hyperplane learned from the SVM model cannot provide a perfect separation. Nonetheless, its combination with the subsequent axis-parallel splits leads to a very good approximation of the optimal solution. The surrogate method 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 axis-parallel splits. First, the 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. 3. 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. 3 ($d_3 > 75.75$) 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 either 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. 4 provides a visualization of the piecewise affine policy of the prescriptive tree with non-orthogonal splits (middle of Fig. 3), focusing on $\mathcal{U}_1, \mathcal{U}_2$. The tree leaves effectively constitute a set of affine policies. The selection of the appropriate policy is communicated to stakeholders in an intuitive manner as it depends on expected line congestion and the net load values at specific nodes, which lead to different generators becoming marginal. For each leaf, we can also infer the impact of a change in the net load in a specific bus on the generation output by checking the linear coefficients. For instance, the root node in Fig. 4 examines if congestion in line 2 is expected; if not, then we evaluate d_3, which checks whether p_2 reaches its maximum capacity. 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 $d_1 = 110$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 (Section V-A), present our main results (Section V-B), and provide additional results under challenging operating conditions (Section V-C). The code to reproduce the results is made available in [32].

A. Experimental Setup

1) Test Cases: The proposed methodology is demonstrated on a range of PGLib-OPF networks v21.07 [31] 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 [31]. 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 axis-parallel splits.
- `APTH`: an affine prescriptive tree using both axis-parallel and non-orthogonal splits.
- `APTH-rlx`: an affine prescriptive tree using both axis-parallel and non-orthogonal 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. Axis-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). 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 [8], [9], [12], 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 [9]. The rest of the hyperparameters are also set according to [9] 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 [33] with default settings. All experiments are run on a standard PC featuring an 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

$$100\times\frac{1}{N_{\text{test}}}\sum_{i\in[N_{\text{test}}]}\frac{\mathbf{c}^\intercal(\hat{\mathbf{p}}_i-\mathbf{p}_i^*)}{\mathbf{c}^\intercal\mathbf{p}_i^*},$$

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. Overall, the SVM classifiers, even though they only utilize a linear kernel, provide very good out-of-sample performance. 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.

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 (%) BEFORE PROJECTION.

	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)**

Table II presents the out-of-sample MCI for the examined test cases. For `NN-prj`, we also report the percentage of infeasible solutions, i.e., the percentage of solutions that require a post hoc projection step to recover a feasible solution for the DC-OPF problem (1). Clearly, tree-based solutions are feasible by design and their infeasibility rate is always zero, thus we omit it from Table II. In almost all cases, the lowest MCI is smaller than 1%, which is on par with previous works and illustrates that ML methods do a very good job at forecasting solutions to the DC-OPF problem. The worst performance is observed for case300, which is probably attributed to the large number of lines facing congestion.

Overall, considering non-orthogonal 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 non-orthogonal splits using 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 splits.

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-

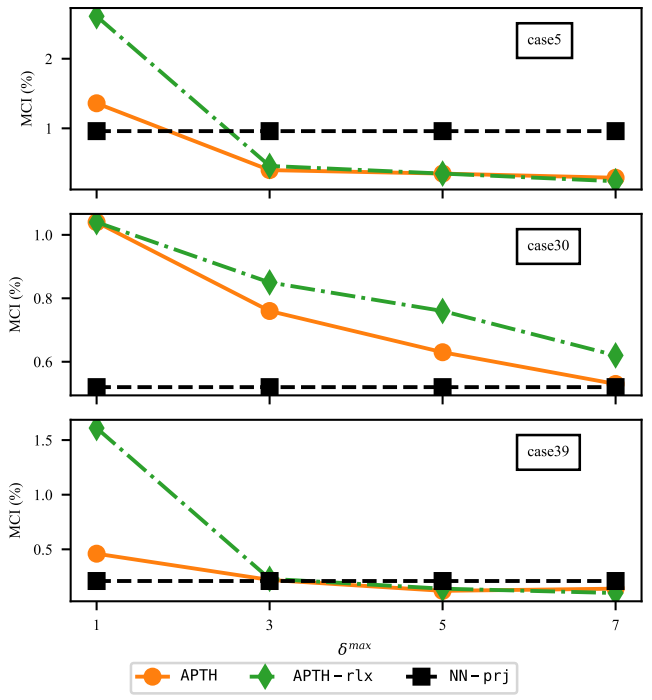


Fig. 5. MCI versus maximum tree depth δ^{max} (uniform uncertainty).

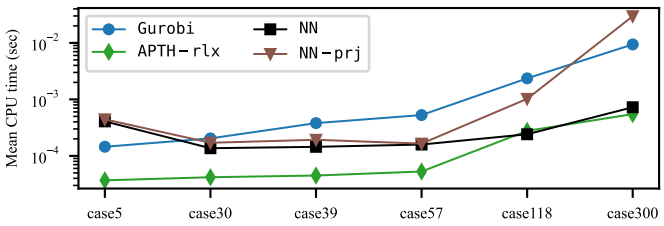


Fig. 6. Mean CPU time to solve a single problem instance. NN denotes the inference time of the NN-based model without projection. The y-axis is in logarithmic scale.

case being observed for case118 and case300, for both types of uncertainty.

We now 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 considers only axis-parallel 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. 5 plots the out-of-sample

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 (%) BEFORE PROJECTION.

	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)**

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. 6 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 of NN−prj is below 10% and a post hoc projection is rarely required, e.g., case30 through case118, the inference time of NN−prj is smaller than that of Gurobi. However, in case300, when the infeasibility rate of NN−prj reaches over 40%, the required projection step to recover a feasible solution negates any improvement in inference speed and leads to higher CPU time than Gurobi. Thus a high infeasibility rate may nullify the intended purpose of applying end-to-end learning in the first place.

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 [31], 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 quite 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, the 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` performs better only when its 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. 6, 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 the feasibility of decisions. We proposed domain-informed, non-orthogonal splits, using a set of hyperplanes to model 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, non-orthogonal splits, which led to an average performance increase of 36% compared to tree models using only axis-parallel splits. Further, shallow prescriptive trees with non-orthogonal splits 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 and using non-linear classifiers that also retain the computational tractability of the proposed policy, e.g., SVMs with a piecewise linear feature mapping. Another interesting direction to explore is considering an online setting and adapting a trained decision tree to deal with changes in grid configuration or a shift in the distribution of net load, e.g., due to increased net load uncertainty. Finally, we aim to extend the proposed method to other variations of the DC-OPF problem, e.g., Security Constrained DC-OPF, as well as other linearized power flow formulations that also consider reactive power and voltage constraints.

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] ENTSO-E, "Options for the design of european electricity markets in 2030." [Online]. Available: https://eepublicdownloads.entsoe.eu/clean-documents/Publications/Market%20Committee%20publications/210331_Market_design%202030.pdf

[3] MISO, "Real-time energy and operating reserve market software formulations and business logic." Business Practices Manual Energy and Operating Reserve Markets Attachment D, 2022.

[4] 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.

[5] W. Chen, M. Tanneau, and P. V. Hentenryck, "End-to-end feasible optimization proxies for large-scale economic dispatch," *IEEE Transactions on Power Systems*, pp. 1–12, 2023.

[6] 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.

[7] M. E. Kaminski, "The right to explanation, explained," *Berkeley Technology Law Journal*, vol. 34, no. 1, pp. 189–218, 2019.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

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

[14] 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.

[15] 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.

[16] 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.

[17] 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.

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

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

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

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

[22] 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.

[23] 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.

[24] 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.

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

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

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

[28] 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.

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

[30] 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.

[31] 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.

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

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

Akylas Stratigakos (Member, IEEE) received a Diploma degree in electrical and computer engineering from the University of Patras, Greece, in 2016 and a Ph.D. degree from Mines Paris, PSL University, Paris, France in 2023. He is currently a postdoctoral researcher at the Department of Electrical and Electronic Engineering at Imperial College London, London, U.K. His research interests include energy forecasting, decision-making under uncertainty, and machine learning applications in power systems.

Salvador Pineda (S'07-M'11-SM'22) received the Ingeniero Industrial degree from the University of Málaga, Spain, in 2006, and a Ph.D. degree in Electrical Engineering from the University of Castilla-La Mancha, Spain, in 2011. He is currently an associate professor in the Department of Electrical Engineering at the University of Málaga in Spain. His research interests are in the fields of power system operation and planning, decision-making under uncertainty, bilevel programming, machine learning, and statistics.

Juan M. Morales (S'07-M'11-SM'16) received the Ingeniero Industrial degree from the University of Málaga, Málaga, Spain, in 2006, and a Ph.D. degree in Electrical Engineering from the University of Castilla-La Mancha, Ciudad Real, Spain, in 2010. He is currently an associate professor in the Department of Statistics and Operations Research at the University of Málaga in Spain. His research interests are in the fields of Data Science and Optimization; decision-making under uncertainty; machine learning, and power systems economics, operations, and planning.

Georges Kariniotakis (Senior Member, IEEE) was born in Athens, Greece. He received the engineering and M.Sc. degrees from Greece, in 1990 and 1992, respectively, and the Ph.D. degree from Ecole des Mines de Paris, Paris, France, in 1996. He is currently a Professor at Mines Paris, PSL University. He is the head of the Renewable Energies and Smart Grids Group at PERSEE Centre. He has authored more than 310 scientific publications in journals and conferences. He has been involved as a participant or coordinator in more than 50 research projects in the fields of renewable energies integration and power systems management and planning. Among them, he was the coordinator of some major EU projects in the field of wind power forecasting, such as Anemos (FP5), Anemos.plus (FP6), and SafeWind (FP7) projects. He is currently the coordinator of the H2020 Smart4RES project on short-term renewable energy forecasting and applications. His scientific interests include, among others, timeseries forecasting, decision-making under uncertainty, modeling, management, and planning of power systems.