

From Optimization-based Machine Learning to Interpretable Security Rules for Operation

Jochen L. Cremer, *Student Member, IEEE*, Ioannis Konstantelos, *Member, IEEE*, and Goran Strbac, *Member, IEEE*,

Abstract—Various supervised machine learning approaches have been used in the past to assess the power system security (also known as reliability). This is typically done by training a classifier on a large number of operating points whose post-fault status (stable or unstable) has been determined via time-domain simulations. The output of this training process can be expressed as a security rule that is used online to classify an operating point. A critical, and little-studied aspect of these approaches is the interpretability of the rules produced. The lack of interpretability is a well-known issue of some machine learning approaches, especially when dealing with difficult classification problems. In the case of the security assessment of the power system, which is a complex mission-critical task, interpretability is a key requirement for the adoption and deployment by operators of these approaches.

In this paper, for the first time, we explore the trade-off between predictive accuracy and interpretability in the context of power system security assessment. We begin by demonstrating how Decision Trees (DTs) can be used to learn data-driven security rules and use the tree depth as a measure for interpretability. We leverage disjunctive programming to formulate novel training methods, capable of learning high-quality DTs while still maintaining interpretability. In particular, we propose two new approaches: (i) Optimal Classification Trees (OCT*) is proposed for training DTs of low-depth and (ii) Greedy Optimization-based Tree (GOT) is proposed for training DTs of intermediate depth, where the increased computational burden is managed by exploiting the nested tree structure. We also demonstrate that the ability to generate high-quality interpretable rules can actually translate to impressive benefits in terms of training requirements. Through case studies on the IEEE 68-bus system, we demonstrate that the proposed methods can produce DTs of higher quality compared to the state-of-the-art approach CART, also if the DT was trained on a significant smaller database, resulting in computational savings of 80 %. Given that generating a large training database is a practical bottleneck in these data-driven approaches, this is a significant breakthrough for real-world application.

Index Terms—Machine Learning, Optimal Classification Trees, Power Systems Operation, Security Rules, Dynamic Stability

NOMENCLATURE

Indices

i	observation
j	feature
m, k	branch node
t	terminal node

Sets

Ω_k^B	branch nodes of the look-ahead DT of node k
Ω_k^E	terminal nodes of the look-ahead DT of node k
Ω^J	features

J. L. Cremer, I. Konstantelos, and G. Strbac are with the Department of Electrical and Electronic Engineering, Imperial College London, London, United Kingdom e-mail: {j.cremer16, i.konstantelos, g.strbac}@imperial.ac.uk

Ω_k^T	training observations following branch node k
$\Omega_{k_L}^B$	branch nodes of the DT following left branch of k
$\Omega_{k_R}^B$	branch nodes of the DT following right branch of k
$\Omega_{k_L}^E$	terminal nodes terminating the left branch of node k
$\Omega_{k_R}^E$	terminal nodes terminating the right branch of k
Ω_k^T	training observations
Ω^{BC}	candidate branch nodes
Ω^B	branch nodes of the global DT
Ω^E	terminal nodes of the global DT

Parameters

$\hat{L}$	base-line
$\mathcal{C}_m^L$	left child node of node m
$\mathcal{C}_m^R$	right child node of node m
$\mathcal{M}^r$	r th big- M value
$\mathcal{N}_m^{\text{con}}$	maximum #features to combine linearly in m
$\mathcal{N}_m^{\text{min}}$	minimal #observations in a terminal node
$\mathcal{P}_m$	parent node of branch node m
$\mathcal{U}_t$	class label of terminal node t
$\mathcal{X}_{ij}$	normalized value of feature j and observation i
$\mathcal{Y}_i$	class label of observation i
ε	small value
$\zeta_k^{\mathcal{T}_L}$	training error corresponding to the left branch of k
$\zeta_k^{\mathcal{T}_R}$	training error corresponding to the right branch of k

Continuous Variables

$\tilde{a}_{jm}$	contribution of feature j in the linear split of m
b_m	threshold of condition for branch node m
l_t	loss of the terminal node t
n_t^1	#observations with class label 1 in terminal node t
n_t^A	#observations of all classes in terminal node t

Binary Variables

a_{jm}	1, if branch node m splits feature j , 0 otherwise
d_m	1, if branch node m is built, 0 otherwise
f_t	1, if observations are in terminal node t , 0 otherwise
y_t	1, if class 1 corresponds to node t , 0 otherwise
z_{ti}	1, if observation i is in terminal node t , 0 otherwise

I. INTRODUCTION

The increase in the complexity of the power system and the massive integration of renewable power generation renders critical the development of new operating tools to manage the security of the system under uncertainty [1]. Given this new reality, a new set of data-driven approaches have emerged to assess the security of the system by deploying statistical (machine) learning into a Monte Carlo framework.

The first step of these data-driven approaches is to fit a statistical model to past historical operating data; the entire procedure is illustrated in Fig. 1. Subsequently, a population

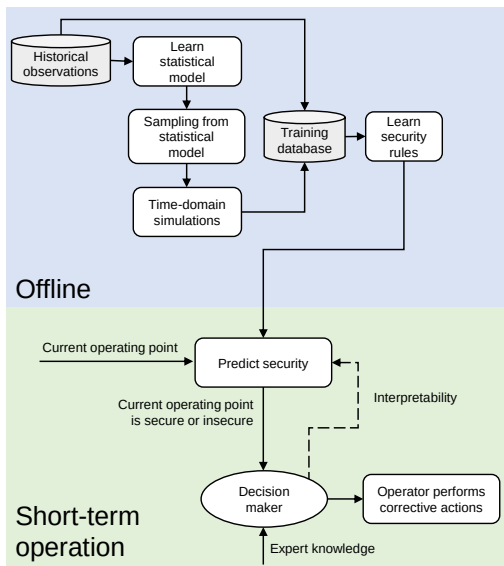


Fig. 1: Data-driven work flow to learn security/reliability rules.

of potential operating points that may occur in the next hours/days is sampled from this model. For each point, a list of contingency scenarios are simulated and the post-fault security is determined. Finally, the pre-fault variables are used as features and the post-fault security as a label to learn a classifier, such as a DT (the security rules), through standard machine learning approaches [2]. The underlying idea is that a Transmission System Operator (TSO) can execute the described training procedure periodically and offline and use the classifiers online to predict the post-fault security of unseen operating points. Thus, the TSO can sample a very large number of operating points that are likely to occur and can avoid time-consuming simulations for these operating points. Such a security assessment can increase the situational awareness [3] of the TSO and may allow operating closer to the system's physical limitations, thereby resulting in cost savings in operation [4].

However, such a trend towards full automation of Dynamic Security Assessment (DSA) is a new development in control rooms. Operators that have relied on manual/visual situational awareness for decades, which involved heuristic selections of simulations and the derivation of heuristic rules from them, have now to start "trusting" this process to an automatic algorithm. Due to the critical nature of the task at hand, it is important that humans are still involved in the control loop and can understand the rules with little inspection. Thus, the ability to interpret and understand security rules is a key requirement of these rules [2]. For this reason, most researchers have studied the use of DTs (e.g., [5]–[9]), and recently fine operating rules [10] as these classifiers offer a high level of interpretability [11]. Some researches have also focused on various other classifiers, such as neural networks [2], deep learning [12], ensemble approaches [13]–[15], such as random forests, or also on clustering approaches, such as k-nearest neighbours [16]. These second set of various classifiers result in better predictive performances, however,

they are typically less interpretable than DTs [11]. For instance ensemble approaches (such as used in [14], [15]) are less interpretable than DTs although they may consist out of DTs themselves. An ensemble of DTs is less interpretable than single DTs as the corresponding DTs may overlap and their classification output may even be in contradiction to each other, as then, the average votes of these DTs are taken as a final classification output; this voting process is not very interpretable.

As mentioned, interpretability is a key requirement of data-driven security assessments as it ensures the ability to understand how a classifier predicts a particular operating point [2], as such, the idea is to learn classifiers that are both, interpretable and accurate in their predictions. This balance was recently investigated by using fine operation rules in [10] and critical flowgates, which were learned from experts knowledge. To learn DTs with high predictive performance and high interpretability requires to balance/trade-off the depth of the DT. Whereas the interpretability decreases with the DT depth, the predictive performance increases (when underfitting) up to a maximal first, and then, decreases again (when overfitting). In this paper, we will show how this trade-off can be improved by using optimization-based learning.

In the majority of the literature, DTs are learned using the Classification and Regression Tree approach (CART) [17]; a forward-selection approach that sequentially grows the DT by making step-wise local decisions based on impurity measures from information theory. In each step, the feature space is split into two half-spaces without considering the impact of future splits in the DT. Consequently, strong split combinations might be hidden behind weak splits. Although CART is one of the most popular algorithms in machine learning due to its speed and scalability, this forward-selection and step-wise nature is the shortcoming of CART and similar approaches and might lead to DTs that do not capture the underlying characteristics of the data well. Researchers have tried to address this shortcoming by either pruning the DT [18] or using lookahead heuristics during DT-learning, such as ID3-k (Iterative Dichotomiser 3) [19]. Lookahead heuristics consider the optimization of deeper DTs instead of a single split and thus, might produce better splits [20]. However, the so-called lookahead pathology of DTs suggests that an increase in lookahead-levels would rather reduce than improve the predictive performance [21]. Several other approaches exist to infer the complete DT in a single step, including the use of mathematical optimization (e.g. by using stochastic gradient descent [22]). Recently, researchers have deployed mixed-integer optimization to learn an OCT in order to account for the underlying nature of the DT problem: a series of discrete decisions [23]. The shortcoming of OCT is the computational limitation with respect to large DTs. However, as outlined, when using the DT for the security assessment of a power system, shallow, and therefore interpretable DTs are preferred.

A. Contributions

For the first time, we explore the trade-off between predictive accuracy and interpretability. We propose the application

of two optimization-based approaches to learning a DT for the security assessment of power systems, which fulfil different requirements in the interpretability of these machine learning based approaches. These two approaches, OCT* and GOT, aim for different DT depths, thus they are not used at the same time. One of the two approaches is selected beforehand depending on the user-specified interpretability in terms of maximal DT depth. Both approaches aim to better trade-off interpretability and predictive performance and provide a highly flexible splitting scheme to learn various DT structures. These include single-variate or multivariate conditions. The first approach OCT* improves OCT and the second approach GOT deploys sequentially a variation of the OCT* approach on subsets of the DT structure. We improve the OCT approach [23] in changing the model of branching conditions. We compute smaller Big- M values and apply disjunctive programming [24]. Smaller Big- M values reduce the feasible region, the search space of the optimization algorithm and in combination with disjunctive programming, these changes enable solvers to make use of branch-and-bound search more effectively in order to achieve superior computational performance. These changes are part of recent algorithmic developments, such as in [25]. Our resulting OCT* approach is suitable for shallow DTs with very small DT depths. For intermediate DT depths, we propose the second, lookahead-based approach GOT, where a variation of OCT* is sequentially applied on subsets of the DT structure. This approach selects heuristically a subset of the DT structure ('looking ahead in the DT structure'), learns locally optimal parameters that describe this subset by applying OCT* while fixing the parts of the structure that have been already learned. In this heuristic search, all possible DT structures are taken as candidates and individually optimized, afterwards the best is selected, which is considered as the local optimal structure.

To study the proposed approaches, an IEEE 68-bus case study is used. First, we show that the existing OCT approach results in a weak computational performance and, subsequently, how our approaches OCT* and GOT address this issue. We then show the inflexibility in the splitting scheme of CART and how the splitting scheme of optimization-based DTs is more flexible and can be used to better capture the underlying characteristic of the data. We continue by studying the predictive performance of the optimization-based DTs and compare it to CART. Moreover, we discuss how much the training database size can be reduced when moving from CART to optimization-based DTs. Lastly, the interpretability in terms of DT depth is discussed as well as the scalability to large grids and the limitations of the optimization-based approaches.

The rest of the paper is structured as follows. In Section II, we present the OCT* approach, focusing on the improvements that we have made. Thereafter, in Section III, the approach to learn a GOT is introduced. Subsequently, the case study is presented in Section IV. Finally, Section V is the conclusion.

II. OPTIMAL CLASSIFICATION TREES

We have improved the OCT approach [23] by adjusting the way branch node conditions are modelled. We refer to

the original approach as OCT and our adjusted approach will be referenced as OCT*. Furthermore, we distinguish between single and multivariate conditions by using (S) and (M), respectively. We start by showing the OCT(S) approach.

A. Current state-of-the-art OCT(S) approach

$$\min \quad \frac{1}{\hat{\mathcal{L}}} \sum_{t \in \underline{\Omega}^E} l_t + \alpha \sum_{m \in \underline{\Omega}^B} d_m \quad (1a)$$

$$\text{s.t.} \quad \sum_{j \in \Omega^J} a_{jm} = d_m \quad \forall m \in \underline{\Omega}^B \quad (1b)$$

$$d_m \leq d_{P_m} \quad \forall m \in \underline{\Omega}^B \setminus \{1\} \quad (1c)$$

$$0 \leq b_m \leq d_m \quad \forall m \in \underline{\Omega}^B \quad (1d)$$

$$a_m^T \mathcal{X}_i \leq b_m + \mathcal{M}^1(1 - z_{ti}) \quad \forall i \in \underline{\Omega}^T, \quad \forall m \in \underline{\Omega}^B, \quad \forall t \in \Omega_m^{E_L} \quad (1e)$$

$$a_m^T \mathcal{X}_i \geq b_m + \epsilon - \mathcal{M}^2(1 - z_{ti}) \quad \forall i \in \underline{\Omega}^T, \quad \forall m \in \underline{\Omega}^B, \quad \forall t \in \Omega_m^{E_R} \quad (1f)$$

$$\sum_{t \in \underline{\Omega}^E} z_{ti} = 1 \quad \forall i \in \underline{\Omega}^T \quad (1g)$$

$$\sum_{i \in \underline{\Omega}^T} z_{ti} \leq f_t \quad \forall t \in \underline{\Omega}^E \quad (1h)$$

$$\sum_{i \in \underline{\Omega}^T} z_{ti} \geq \mathcal{N}^{\min} f_t \quad \forall t \in \underline{\Omega}^E \quad (1i)$$

$$n_t^1 = \sum_{i \in \underline{\Omega}^T} \mathcal{Y}_i z_{ti} \quad \forall t \in \underline{\Omega}^E \quad (1j)$$

$$n_t^A = \sum_{i \in \underline{\Omega}^T} z_{ti} \quad \forall t \in \underline{\Omega}^E \quad (1k)$$

$$l_t \geq n_t^A - n_t^1 - \mathcal{M}^3(1 - y_t) \quad \forall t \in \underline{\Omega}^E \quad (1l)$$

$$l_t \geq n_t^1 - \mathcal{M}^3 y_t \quad \forall t \in \underline{\Omega}^E \quad (1m)$$

$$l_t \leq n_t^A - n_t^1 + \mathcal{M}^3 y_t \quad \forall t \in \underline{\Omega}^E \quad (1n)$$

$$l_t \leq n_t^1 + \mathcal{M}^3(1 - y_t) \quad \forall t \in \underline{\Omega}^E \quad (1o)$$

The optimization with the objective function Eq. (1a) aims to maximize the predictive performance of the classifier (in terms of test accuracy). As such, the objective function balances training error and model complexity. The training error is expressed as the losses l_t across all terminal nodes t , which is normalized to the base-line misclassification training accuracy $\hat{\mathcal{L}}$. $\hat{\mathcal{L}}$ is computed by predicting the most popular class of the training observations $\underline{\Omega}^T$ and computing the number of misclassified observations. The model complexity is expressed as the number of branch nodes built. This second term is also called regularization term in order to regulate the model complexity and to avoid over-fitting. In due course of the MILP optimization procedure, these terms are minimized and the procedure stops either if a user-specified stopping criterion applies (such as restricting the solver time) or if a globally optimal solution is found. The constraints Eqs. (1b) - (1f) are for the branch conditions, Eq. (1g) to ensure the unique

mapping of observations to terminal nodes and Eqs. (1h) - (1o) are for the terminal nodes. The losses l_t of the corresponding terminal nodes are calculated in the optimization as they are optimization variables. These losses are computed in the inequality constraints Eqs. (1l) - (1o), which are big- M constraints and, as such, each constraint models two different equality constraints dependent on the binary variable y_t .

Eqs. (1e) and (1f) are the constraints we will modify in Section II-B. Eq. (1e) corresponds to the case if the node $t \in \Omega_m^{EL}$ terminates in the left branch of m and Eq. (1f) to the case if node $t \in \Omega_m^{ER}$ terminates in the right branch of m . $\mathcal{X}_i$ is the normalized feature vector of observation i , $\mathcal{M}^1$ and $\mathcal{M}^2$ are big- M values. Binary decision variables are the vector a_m^T to denote which feature j is selected for the split in branch node m and z_{ti} to denote if the observation i lies in terminal node t . The threshold of the split that corresponds to m is b_m . The final learned security/reliability rules are defined by a_m^T and b_m .

The downside of Eqs. (1e) and (1f) is their implication on the computational complexity of the resulting MILP. They encode a high number of constraints and a large feasible region. The number of constraints might be high as the problem grows linearly with the number of branch nodes m , the number of terminal nodes t and with the number of observations i . The size of the feasible region might be large as it is sensitive to the selection of the big- M values $\mathcal{M}^1$ and $\mathcal{M}^2$. The higher the big- M values, the greater the feasible region. Consequently, small big- M values are preferred as has been recognized in [23]. As such, they were selected at $\mathcal{M}^1 = 1$ and $\mathcal{M}^2 = 1$ as all variables are bounded in $[0, 1]$. However, as illustrated in Fig. 2, the bounds of the threshold b_m can be smaller than $[0, 1]$ and, consequently, the feasible region of the constraints Eqs. (1e) and (1f) may be overestimated. The scale of this overestimation depends on the selection of the minimal number of observations in each terminal node $\mathcal{N}^{\min}$ and on the distribution of observations.

B. Proposed OCT*(S) approach

To address the downside described above we have replaced Eqs. (1e) and (1f) with our constraints

$$a_m^T \mathcal{X}_i \leq b_m + \mathcal{M}_i^1 (1 - \sum_{t \in \Omega_m^{EL}} z_{ti}) \quad \forall i \in \underline{\Omega}^T, \forall m \in \underline{\Omega}^B \quad (2a)$$

$$a_m^T \mathcal{X}_i \geq b_m - \mathcal{M}_i^2 (1 - \sum_{t \in \Omega_m^{ER}} z_{ti}) \quad \forall i \in \underline{\Omega}^T, \forall m \in \underline{\Omega}^B \quad (2b)$$

where we have made the following two changes:

a) *Disjunctive formulation*: In Eqs. (1e) and (1f), the branch condition is individually accounted as one constraint for each terminal node t , whereas in our Eqs. (2a) and (2b) a single disjunction is used across all nodes that terminate the branches. This disjunctive formulation reduces the number of constraints; in contrast, the individual constraint involves more variables and a reduction of computational complexity cannot immediately be concluded and is tested in the case study section.

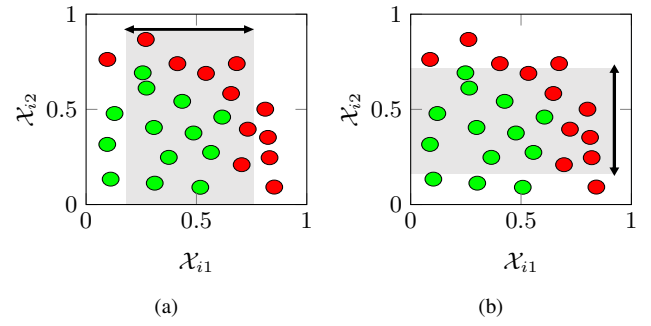


Fig. 2: Feasible region of the threshold variable b_m considering when $\mathcal{N}^{\min} = 4$ minimal observations in each terminal node. $\mathcal{N}^{\min}$ includes both, stable (●) and unstable (●) observations. (a) corresponds to a split in feature $j = 1$ and (b) to a split in feature $j = 2$.

b) *Optimal big- M values*: The big- M values are generalized to the value of 1 in Eqs. (1e) and (1f), resulting in over-estimations of the feasible region of the constraints. In Eqs. (2a) and (2b) optimal big- M values are used for each observation i . To compute these optimal big- M values, we start by sorting the values of the feature matrix for each feature j , where $\mathcal{X}_j^{(u)}$ is the u th largest value in the j th feature. Subsequently, as illustrated in Fig. 2, we can compute bounds for the variable b_m as $\min\{\mathcal{X}_j^{(|\underline{\Omega}^T| - \mathcal{N}^{\min})} \mid \forall j \in \Omega^J\} \leq b_m \leq \max\{\mathcal{X}_j^{(\mathcal{N}^{\min})} \mid \forall j \in \Omega^J\}$, in which the symbol $|\cdot|$ denotes to the cardinality of the set. Note that the upper bound can also be used to tighten the bounding of the variable itself

$$0 \leq b_m \leq \max\{\mathcal{X}_j^{(\mathcal{N}^{\min})} \mid \forall j \in \Omega^J\} d_m \quad \forall m \in \underline{\Omega}^B. \quad (3)$$

The lower bound cannot be used as the underlying logic requires that b_m can be zero. Subsequently, we realize that the minimum and maximum values of the left-hand side of Eqs. (2a) and (2b) for each observation i are $\min\{\mathcal{X}_{ij} \mid \forall j \in \Omega^J\}$ and $\max\{\mathcal{X}_{ij} \mid \forall j \in \Omega^J\}$, respectively. Based on the tight bounds of b_m and the left-hand sides of Eqs. (2a) and (2b), we can compute the optimal big- M values, which are used in our Eqs. (2a) and (2b) as

$$\mathcal{M}_i^1 = \max\{\mathcal{X}_{ij} \mid \forall j \in \Omega^J\} - \min\{\mathcal{X}_j^{(|\underline{\Omega}^T| - \mathcal{N}^{\min})} \mid \forall j \in \Omega^J\} \quad \forall i \in \underline{\Omega}^T \quad (4a)$$

$$\mathcal{M}_i^2 = \min\{\mathcal{X}_{ij} \mid \forall j \in \Omega^J\} - \max\{\mathcal{X}_j^{(\mathcal{N}^{\min})} \mid \forall j \in \Omega^J\} - \varepsilon \quad \forall i \in \underline{\Omega}^T. \quad (4b)$$

C. Flexibility through multivariate conditions (M)

A few adjustments must be made when extending to multivariate conditions OCT*(M). To allow for a maximum number of features $\mathcal{N}_m^{\text{con}} > 1$ in the split of node m , the inequality constraint

$$\sum_{j \in \Omega^J} a_{jm} \leq \mathcal{N}_m^{\text{con}} d_m, \quad \forall m \in \underline{\Omega}^B \quad (5)$$

is used to replace Eq. (1b). Further, as we are moving from a single-variate splitting scheme to a linear (multivariate) splitting scheme, the continuous variable $-1 \leq \tilde{a}_{jm} \leq 1$ is introduced to indicate the level of contribution of each feature

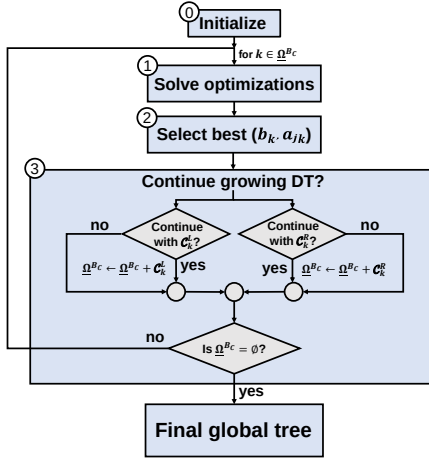


Fig. 3: Procedure to learn a Greedy Optimization-based Tree.

j in the linear combination of the split corresponding to m . As the level of contribution $\tilde{a}_{jm}$ must be zero if a_{jm} is zero, we use

$$-a_{jm} \leq \tilde{a}_{jm} \leq a_{jm} \quad \forall j \in \Omega^J, \forall m \in \Omega^B. \quad (6)$$

Finally, we replace Eq. (2a) and Eq. (2b) with

$$\tilde{a}_m^T \mathcal{X}_i \leq b_m + \mathcal{M}_i^1 (1 - \sum_{t \in \Omega_m^{EL}} z_{ti}) \quad \forall i, \forall m \in \Omega^B \quad (7a)$$

$$\tilde{a}_m^T \mathcal{X}_i \geq b_m + \varepsilon - \mathcal{M}_i^2 (1 - \sum_{t \in \Omega_m^{ER}} z_{ti}) \quad \forall i, \forall m \in \Omega^B \quad (7b)$$

where the linear combination of the contributions $\tilde{a}_{jm}$ is used.

III. GREEDY OPTIMIZATION-BASED TREE (GOT)

We present the multivariate GOT(M) as it generalizes to the single-variate case GOT(S). The principle idea of GOT(M) is to sequentially grow the DT in a top-down fashion, as most of the DT-learning approaches do. However, there are two main differences between GOT(M) and most of the DT-learning approaches. (i) GOT(M) can account for a subset of the tree structure in each sequence, whereas other approaches account only for the single splits in each sequence. (ii) The GOT(M) approach uses optimization, a variation of OCT*(M), to infer the split conditions, other approaches use metrics from information theory to make local split decisions, such as in CART. The sequence that the GOT(M) approach performs is illustrated in Fig. 3. Step 0 is to initialize, Step 1 to solve a set of optimizations, each for one candidate split of a single branch node, Step 2 to select the best candidate and Step 3 is the stopping condition of the approach. Each step is described in more detail as follows.

Step 0: Initialize

We start by initializing global sets that overspan the whole DT growing procedure. The symbol $_$ is used to indicate that a set is global. We initialize the set of candidate branch nodes $\Omega^{BC} = \{1\}$. The rest of the sets describing the global DT structure, $\Omega^B = \{\}$ and $\Omega^T = \{\}$, are initially empty. Further, the training set $\Omega_1^T = \Omega^T$ of the first branch node $k = 1$ contains all available training observations.

Step 1: Solve optimizations

In Step 1, a set of optimizations is solved to decide on the branching condition of one candidate branch node $k \in \Omega^{BC}$. As GOT(M) decides on k by accounting for the DT structure that surrounds k , the optimizations consider each a different DT structure. The considered DT structure is user-defined in terms of number of levels to 'look ahead' into the DT depth. Dependent on the number of levels to 'look ahead', the optimizations account for a set of branching nodes Ω_k^B and terminal nodes Ω_k^E . Across the optimizations, we are varying the combination of class labels $\mathcal{U}_t$ of the terminal nodes $t \in \Omega_k^E$. E.g., if the user decides that GOT(M) may look one level ahead, the static DT structure is described by the sets $\Omega_k^B = \{k, m1, m2\}$, $\Omega_k^E = \{t1, t2, t3, t4\}$, $\Omega_k^{EL} = \{t1, t2\}$, $\Omega_k^{ER} = \{t3, t4\}$, $\Omega_{m1}^{EL} = \{t1\}$, $\Omega_{m1}^{ER} = \{t2\}$, $\Omega_{m2}^{EL} = \{t3\}$ and $\Omega_{m2}^{ER} = \{t4\}$. Each possible combination of the parameter $\mathcal{U}_t$ must be considered in separate optimizations, except $\mathcal{U}_{t1} \neq \mathcal{U}_{t2}$ and $\mathcal{U}_{t3} \neq \mathcal{U}_{t4}$, resulting, in this case, in 4 optimizations. These combinations are exempt as two neighboring terminal nodes would not entail the same class at the same time. Each optimization is formulated as follows

$$\min \sum_{i \in \Omega_k^T} \sum_{t \in \Omega_k^E} (\mathcal{Y}_i + \mathcal{U}_t - 2\mathcal{Y}_i \mathcal{U}_t) z_{ti} \quad (8a)$$

$$\text{s.t.} \quad \sum_j a_{jm} = \mathcal{N}_m^{\text{con}} \quad \forall m \in \Omega_k^B \quad (8b)$$

$$-a_{jm} \leq \tilde{a}_{jm} \leq a_{jm} \quad \forall j \in \Omega^J, \forall m \in \Omega_k^B \quad (8c)$$

$$\sum_{t \in \Omega_k^E} z_{ti} = 1 \quad \forall i \in \Omega_k^T \quad (8d)$$

$$\begin{aligned} \tilde{a}_m^T \mathcal{X}_i &\leq b_m \\ &+ \mathcal{M}_i^1 (1 - \sum_{t \in \Omega_m^{EL}} z_{ti}) \quad \forall i \in \Omega_k^T, \forall m \in \Omega_k^B \end{aligned} \quad (8e)$$

$$\begin{aligned} \tilde{a}_m^T \mathcal{X}_i &\geq b_m + \epsilon \\ &- \mathcal{M}_i^2 (1 - \sum_{t \in \Omega_m^{ER}} z_{ti}) \quad \forall i \in \Omega_k^T, \forall m \in \Omega_k^B \end{aligned} \quad (8f)$$

where $\Omega_m^E = \Omega_m^{EL} \cup \Omega_m^{ER}$. The objective function Eq. (8a) is the training error across all training observations in Ω_k^T . Most of the constraints have been explained in Section II; however, it is worth mentioning that Eq. (8b) is here an equality constraint and the set Ω_k^T is specific for the branch node k , which is discussed further in Step 3.

Step 2: Select the best structure

After solving all optimizations, the best DT structure that surrounds the branch node condition k and the best combination of $\mathcal{U}_t$ is selected by comparing the objective values, the training error, across the optimizations. The structure with the lowest objective value is selected. We use the result of the winning optimization and store the corresponding vectors a_{jk} , $\tilde{a}_{jk}$, $\mathcal{U}_t$ and the scalar value b_k for the global DT. As we have assessed k as a candidate branch node we can add k to the global branch nodes $\Omega^B \leftarrow \Omega^B \cup \{k\}$ and remove it from the candidates $\Omega^{BC} \leftarrow \Omega^{BC} \setminus \{k\}$.

Step 3: Continue growing DT?

In Step 3, the performance of the direct child nodes of the winning DT structure is analyzed and it is determined, as illustrated in Fig. 3, if the approach terminates in their corresponding branches. To analyze the performance of the child nodes, we start by computing their shares of the training error

$$\zeta_{k_L}^T = \sum_{i \in \Omega_k^T} \sum_{t \in \Omega_k^{E_L}} (\mathcal{Y}_i + \mathcal{U}_t - 2\mathcal{Y}_i \mathcal{U}_t) z_{ti} \quad (9a)$$

$$\zeta_{k_R}^T = \sum_{i \in \Omega_k^T} \sum_{t \in \Omega_k^{E_R}} (\mathcal{Y}_i + \mathcal{U}_t - 2\mathcal{Y}_i \mathcal{U}_t) z_{ti}. \quad (9b)$$

Subsequently, if $k > 1$ we compare this training error to the training error of the parent node $\mathcal{P}_k$. If the training error in the left branch has reduced $\zeta_{k_L}^T < \zeta_{\mathcal{P}_k}^T$, then we continue growing the DT. This decision involves to adding the child node to the candidates $\underline{\Omega}^{BC} \leftarrow \underline{\Omega}^{BC} \cup \{\mathcal{C}_k^L\}$ and computing the training set $\Omega_{\mathcal{C}_k^L}^T = \{i \in \Omega_k^T \mid z_{ti} = 1 \wedge t \in \Omega_k^{E_L}\}$. However, if $\zeta_{k_L}^T \geq \zeta_{\mathcal{P}_k}^T$, the DT-growing terminates in this branch. This condition is one of the rules to stop growing the DT. The second condition to stop growing the DT is if the user-specified maximal DT depth is reached. In case of stopping DT growth, the sets of the DT structure in this branch, Ω_k^{BL} and Ω_k^{EL} , are added to the global sets $\underline{\Omega}^B \leftarrow \underline{\Omega}^B \cup \Omega_k^{BL}$ and $\underline{\Omega}^T \leftarrow \underline{\Omega}^T \cup \Omega_k^{EL}$, respectively. Accordingly, we perform the equivalent logic for the right child $\mathcal{C}_k^R$: if $\zeta_{k_R}^T < \zeta_{\mathcal{P}_k}^T$ then update $\underline{\Omega}^{BC}$ and $\Omega_{\mathcal{C}_k^R}^T$ else if $\zeta_{k_R}^T \geq \zeta_{\mathcal{P}_k}^T$ then update $\underline{\Omega}^B$ and $\underline{\Omega}^T$. Finally, if $\underline{\Omega}^{BC} \neq \emptyset$, then we continue with Step 1 to assess the next candidate branch node. If no candidates are left $\underline{\Omega}^{BC} = \emptyset$, the approach terminates and the global DT structure is defined by the sets $\underline{\Omega}^B$, $\underline{\Omega}^T$ and the parameters a_{jm} , $\tilde{a}_{jm}$ and $b_m \forall m \in \underline{\Omega}^B$ and $\mathcal{U}_t \forall t \in \underline{\Omega}^T$.

In terms of convergence of the GOT(M) approach, the DT is grown further iff a child reduces the training error by predicting at least one more training observation $i \in \Omega^T$ correct (note the strict conditions $\zeta_{k_L}^T < \zeta_{\mathcal{P}_k}^T$ and $\zeta_{k_R}^T < \zeta_{\mathcal{P}_k}^T$). Given this and the definiteness of the size of the training dataset $|\Omega^T| \in \mathbb{N}$, the training error decreases with further progression of growing the DT, converging to zero if no other stopping criterion holds before (such as maximal DT depth). Therefore, the GOT(M) approach is guaranteed to converge.

IV. CASE STUDY

With the use of the IEEE 68-bus system, we compare both optimization-based approaches OCT* and GOT against CART in terms of computational performance, flexibility in the splitting scheme, predictive performance, training database sizes, the interpretability, the computational capability for large systems, and finally a discussion of the limitations.

A. Test system and assumptions

The IEEE 68-bus system was used and all data was taken from [26]. A set of 9000 observations was sampled, where each observation described a steady-state operation of the system under AC assumptions. The 235 features, which described

the steady-state of each observation i , were the active and reactive power injections (generation and load), the voltages, and the phase angles of each bus which were compressed to $\mathcal{X}_{ij}$. The transient stabilities of three-phase faults on a set of 22 lines were simulated in the time-domain for all of the steady-state observations. If the phase angles and internal voltages of all generators remained within the corresponding operating limits for the considered simulation time interval, the steady-state observation i was assumed stable ($\mathcal{Y}_{ic} = 1$), otherwise, unstable ($\mathcal{Y}_{ic} = 0$) for the line contingency c . Note, we have added the index c to $\mathcal{Y}_{ic}$ in order to denote that we have used 22 different labels for each observation i .

In order to create steady-state observations with high variability, the active load observations were drawn from a multivariate Gaussian distribution and a Pearson's correlation coefficient of 0.75 was used between all load pairs. Subsequently, the loads were converted to a marginal Kumaraswamy(1.6, 2.8) distribution by making use of the inverse transformation method, then scaled to be within $\pm 50\%$ of the nominal active loads. The reactive loads were scaled from the nominal value by assuming constant impedances at the buses. The shares of the active power of all generators were uniformly distributed, then normalized with respect to their capacity-sharing ratio. The reactive powers were distributed in the same way. Finally, to achieve feasibility of each observation and to constrain the power factors within $[0.75, 1]$, an optimization was solved accounting for the AC network model. The objective function was the minimization of all absolute deviations from the active and reactive power shares, and loads were parameters. To solve these nonlinear optimizations, IPOPT 3.12.4 [27] was used. In the resulting 22 datasets, the ratio of stable to unstable observations was between 36% and 69% with 52% as mean.

The data was pre-processed by applying a feature selection step to reduce the computational complexity for the discussed DT approaches. The top 2 (or 4) features for each dataset were selected based on their gini importance, the normalized total reduction of the gini impurity (computed by using CART).

To study the DT learning approaches, the training/testing split was 70%/30% in all studies. The training and testing populations were drawn from the same probability distribution. Changes in the probability distributions between training and testing were not considered. Also, one DT model of the corresponding learning approaches must be learned for each of the 22 datasets.

To study CART [17] we used the package *scikit-learn* 0.18.1 [28] in Python 3.5.2. The most of the hyperparameters and settings were used from the default settings of *scikit-learn*, such as the minimization of gini impurity, choosing the best split, minimal observations to split an internal node was 2, all features were used when looking for the best split, the minimal decrease in gini impurity that a node splits was 0, no class weights were used and the data was not presorted. The hyperparameters that were varied as part of our study were the DT depth and the minimal number of observations per terminal node $\mathcal{N}^{\min}$. In the OCT [23] and our proposed OCT* approach, we set the hyperparameter $\alpha = 0$, as we were directly controlling the model complexity by limiting the maximum DT depth. Furthermore, the hyperparameter

TABLE I: Computational impact of our improvements to OCT. The computational times is an average across learning 10 DTs of the same dataset.

Approach	#constraints	#continuous variables	#binary variables	solver time
OCT(S)	10957	20	3377	94(50) s
OCT(S) + disjuncts	9277	20	3377	77(38) s
OCT(S)+ big- M	10957	20	3377	69(14) s
OCT*(S)	9277	20	3377	63(19) s

TABLE II: Comparison of the average computational times across all 22 datasets of all DT learning approaches for learning the DT (pure solver time of optimization-based approaches) and predicting an observation.

Approach	OCT(S)	OCT*(S)	GOT(S) look-ahead 0	GOT(S) look-ahead 1	CART
DT learning	517(403) s	306(237) s	7(1) s	172(92) s	< 1 s
Prediction	< 0.1 s	< 0.1 s	< 0.1 s	< 0.1 s	< 0.1 s

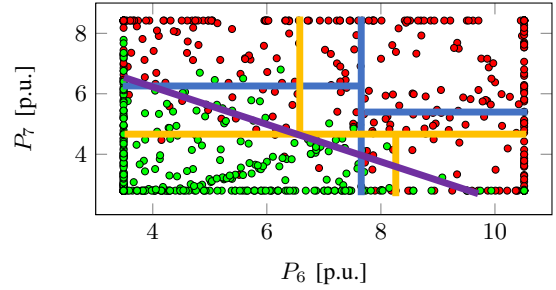
$\varepsilon = 0.001$ in GOT, OCT, and OCT* and all MILPs were implemented using the package Pyomo 5.2 [29] in Python 3.5.2. CPLEX 12.7 was used to solve the MILPs, all to global optimality, on a standard desktop machine. Thus, in the OCT and OCT* approaches, it was guaranteed that the best DT subject to the specification of the optimization problems was found.

B. Computational performance of learning a DT

The computational performance of using a DT for testing (predicting an observation) was negligible and similar for different DT approaches. Thus, the following two studies focused on the computational performance of the DT learning procedure when using different approaches.

The first study focused on the computational performances of OCT*(S) and OCT(S). The problem investigated involved $|\underline{\Omega}^T| = 840$ training observations of one contingency c and was reduced to account for the 2 features with the highest gini importance. The minimal number of observations corresponding to each terminal node was $\mathcal{N}^{\min} = 10$ and the maximum DT depth was $D = 2$. Subsequently, 10 different training/testing set combinations were analyzed, as such, 10 different DTs were computed for each approach. To show the effectiveness of both our changes, (i) the disjunctive reformulation and (ii) the optimal calculation of big- M values, the pure solver times were computed individually based on these changes. The results are summarized in Table I. Whereas the computational complexity in terms of the number of constraints reduced from OCT(S) to OCT*(S), the number of variables did not decrease. In terms of solver time, each of the changes that we made to OCT(S) resulted individually in reductions. When applying both changes at the same time, an average-absolute reduction of 31 s (from 94 s to 63 s), a relative reduction of 33 %, was obtained.

In a second study, the computational performance of all discussed approaches was analyzed. The complexity of the problem was slightly changed: the top 4 features, $|\underline{\Omega}^T| = 560$ training observations and $\mathcal{N}^{\min} = 10$ were used. The maximum DT depth was $D = 2$ in OCT(S) and OCT*(S) and

Fig. 4: Flexibility of splitting schemes in the domain of active generator powers P_6 and P_7 for contingency on the line connecting buses 23 and 24. The schemes were learned by CART (—), OCT*(S) (—) and OCT*(M) (—). Stable (●) and unstable (●) observations (training and testing) are shown.

$D = 5$ in GOT(S), as GOT(S) is to be used in intermediate DT depths. The computational time was computed across all 22 datasets, 10 times each, each time with different sets of training and testing observations (220 DTs in total). The computational times were exclusively the pure solver times (in GOT(S), the total across all optimizations). The mean and standard deviations of the computational times are reported in Table II. As before, the OCT*(S) outperformed OCT(S), in this study by a relative reduction in computational time of 41 %. The results of GOT(S) with looking ahead 0 and 1 level into the DT were 7(1) s and 172(92) s, respectively. In terms of the time to predict an observation, as mentioned, this time is low for all DTs. There are no significant differences in the approaches, as it is in all DT approaches a matter of implementation and, in the end, a simple check of a set of linear logical conditions. Recall, Table II presents the mean of 220 DTs, computed across very different datasets, which explains the high standard deviation.

C. Flexibility in the splitting scheme

Optimization-based DTs allow, in a straightforward manner, for multivariate splits OCT*(M), which increases the flexibility in the splitting scheme. To illustrate the importance of this flexibility, the dataset of 800 observations is visualized in Fig. 4, where the binary label of transient stability was the one from the line connecting buses 23 and 24, the same as analyzed before in the first study of Section IV-B. The two features identified with highest gini importance P_6 and P_7 were active powers of generators, and the DT learning approaches CART, OCT*(S) and OCT*(M) were constrained to use only these features, along with $\mathcal{N}^{\min} = 10$ and $|\underline{\Omega}^T| = 560$, as before. Whereas CART and OCT*(S) allow for single-variate splits, OCT*(M) allows for multivariate splits (here in two features). To compensate for this difference in the number of variants to-split-in, the approaches were constrained to different tree depths D : CART and OCT*(S) to $D = 2$ and OCT*(M) to $D = 1$. In CART and other tree learning algorithms, strong split combinations might be hidden behind weak splits, as mentioned in Section I. In this example, CART and OCT*(S) found different split combinations. Whereas the initial branch node in CART always branched in P_6 , OCT*(S) always branched in P_7 . In other words, CART was not able to identify that a better split combination might exist, which OCT*(S)

identified. However, it can be seen in Fig. 4 that OCT*(M) can best capture the underlying characteristics of the data, which is also verified by the corresponding predictive performances of these DTs. Based on 240 testing observations, the testing accuracies were at 89 %, 91 % and 95 % for CART, OCT*(S) and OCT*(M), respectively.

D. Predictive performance

To test the predictive performance, two studies with different maximum DT depths were performed. The test accuracy, F_1 score, precision, and recall were computed.

In an initial study, GOT(S) and OCT*(S) were compared against CART by using a maximum DT depth of $D = 2$. The GOT(S) approach was applied in two different variations, with looking zero levels and one level into the DT, respectively. Note that the look-ahead 1 variation is a decomposition of the OCT*(S) problem if $D = 2$. For each dataset 10 DTs were computed using $|\underline{\Omega}^T| = 560$, $\mathcal{N}^{\min} = 1$ and the top 4 features. All metrics were computed for all datasets and all DTs. Each of the resulting 220 DTs was assessed with a different testset containing 240 observations. Their means and standard deviations are reported in Table III. The test accuracies showed an absolute difference of 1 % when using OCT*(S) over CART, which was also the result of the study in [23]. This absolute difference varied between -0.42 % and 6.2 % meaning that in some cases, the improvement was significant. In addition, the absolute number of wins were computed. The optimization-based DTs had 21 times (out of the 22 datasets) the highest test accuracy, therefore, won 21 times. The precision of the optimization-based DTs was lower than CART and in terms of recall, the OCT*(S) approach significantly outperformed CART by an average of 3 %. The F_1 score showed an improvement of 1 % of OCT*(S) over CART. In this study, which was on a low DT depth of $D = 2$, OCT*(S) did in-average outperform CART and GOT(S) did not outperform in-average CART. This was expected, as we proposed OCT*(S) for low DT depths and GOT(S) for intermediate DT depths, therefore, we continued with a study on these DT depths as follows. There were also cases where GOT(S) outperformed OCT*(S), which was caused by the fact that the training error was minimized in the two of the approaches rather than optimizing for one of the testing metrics, which were used for comparing the approaches. As OCT*(S) considers more possible DT structures, the training errors are always lower or equal than in the GOT(S) approaches. However, a lower training error of OCT*(S) did not necessarily imply that the test metrics were also higher. This cannot be implied as the testing metrics and the training accuracy refer to different populations of observations, the training dataset and the testing dataset, respectively.

In a second study, a maximum DT depth of $D = 5$ was used. GOT(S) and CART DTs were compared; OCT*(S) was not included in this study as it did not lead to promising results within reasonable solver-times (in the order of hours) as the duality gap was not sufficiently small. This was expected as OCT*(S) is to be used for very low DT depths, such as 1, 2 or 3 and GOT(S) is for intermediate DT depths, such as 4,

TABLE III: Average predictive performance of DTs with depth $D = 2$ learned by CART, GOT(S) and OCT*(S) approaches across all 22 datasets.

Approach	CART	GOT(S) look-ahead 0	GOT(S) look-ahead 1	OCT*(S)
#wins out of 22	1	6	10	5
Test accuracy	90.1(5) %	90.7(4) %	91.2(4) %	91.1(4) %
Precision	90.5(5) %	90.7(5) %	88.2(6) %	89.3(6) %
Recall	89.2(7) %	89.1(7) %	90.9(6) %	92.1(5) %
F_1 score	89.6(6) %	89.7(6) %	89.4(6) %	90.6(5) %

TABLE IV: Average predictive performance of DTs with maximum depth $D = 5$ learned by CART and GOT(S) approaches across all 22 datasets.

Approach	CART	GOT(S) look-ahead 0	GOT(S) look-ahead 1
#wins out of 22	0	6	16
Test accuracy	90.8(4) %	91.3(4) %	91.8(4) %
Precision	90.2(5) %	93.6(3) %	91.1(4) %
Recall	91.0(5) %	88.0(8) %	92.1(5) %
F_1 score	90.6(5) %	90.5(5) %	91.5(5) %

5, ... The results, presented in Table IV, showed that both GOT(S) cases, with 0 and 1 look-ahead levels outperformed CART by 1 % in terms of test accuracy. In terms of precision and recall, GOT(S) look-ahead 1 showed a slight improvement (over CART) of 1 % and GOT(S) look-ahead 0 showed an improvement in precision and decline in the recall. In terms of F_1 score, only GOT(S) look-ahead 1 outperformed CART by 1 %. In conclusion of both studies, OCT*(S) and GOT(S) had better predictive performances than CART, for small and intermediate DT depths, respectively.

E. Size of the training database

In this study, we evaluated the impact on the size of the training database. We computed how many observations would be needed to learn a DT with CART that had the same predictive performance as an optimization-based DT. The rationale was that this difference in training database sizes informed how much simulation data could be saved when moving from CART to OCT*(S) or GOT(S). This comparison was done for all contingencies and one of them was selected to illustrate the procedure in Fig. 5. For the data corresponding to the contingency on the line connecting buses 24 and 25, the test accuracy was computed when using different sizes of the training database $|\underline{\Omega}^T| = 500, 600, \dots, 6000$. This range was reasonable to study as, for example, the study in [9] used 7160 observations. Note, as before, that each point was an average of 10 different training/test data combinations. Then a standard linear regression curve was fitted to this average test accuracy by minimizing the mean squared error. Subsequently, the test accuracy of OCT*(S) was used to interpolate on this fitted curve. This interpolated value was an estimate of the required training database size when learning a CART DT with similar test accuracy. In the contingency that was used in Fig. 5 the reduction was estimated to be from 3650 to 560 observations, such the training database size was reduced by 85 %. When all 22 contingencies were analyzed in the same way, the outcome was: (i) In 10 contingencies, the increase

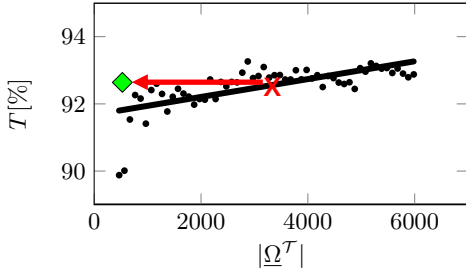


Fig. 5: Less data is required when learning an OCT*(S) (◆) instead of CART DTs (●). CART DTs have been learned for different training database sizes $|\Omega^T|$. By using a linear regression curve (—), the CART DT (X) with equivalent test accuracy T has been identified. Each data point corresponds to the average of 10 DTs learned with different observations.

in the training database size (of up to $|\Omega^T| = 6000$) did not lead to improvements in the test accuracy: the slope of the fitted curve was nearly zero or even negative. (ii) In 9 contingencies, an estimation would have required extrapolation from the CART data of $|\Omega^T| \gg 6000$. These cases were not included as they resulted in very high and unrealistic estimates. (iii) In the remaining 3 contingencies, the mean of the interpolation was 2898 observations. This last outcome in (iii) corresponded to an average reduction of 80 % training observations.

F. Interpretability

To evaluate the impact on the interpretability when using optimization-based DTs we compared the interpretability of our OCT*(S) DTs to the corresponding DTs from CART with different DT depths for all the 22 contingencies. The result and the procedure-used are presented in Fig. 6 for a single contingency on the line connecting buses 28 and 29. For both GOT(S) and CART approaches the maximum DT depth was varied within $D = 1, 2, \dots, 9$ and the average test accuracy was computed using 10 combinations of training and testing set. Note, the actual DT depth could have been smaller, as stopping criteria were applied during DT growth. As before $|\Omega^T| = 560$ training and 240 testing observations were used. The DT with the desired trade-off between predictive accuracy and interpretability was selected to be at a depth of $D = 2$. Subsequently, the optimal OCT*(S) DT corresponding to this trade-off was learned. Then, the CART DT with minimal average actual depth and higher testing accuracy than the OCT*(S) DT was found. E.g., in Fig. 6 this was at $D = 6$. This was also the point with highest predictive accuracy for CART. In this study, no interpolation techniques were applied as the curve shape varied from contingency to contingency. The curve shape varied as it involved different impacts of the under- and overfitting problem (e.g., in Fig. 6 $D < 6$ and $D > 6$ correspond to underfitting and overfitting, respectively). As mentioned, this procedure was repeated for all 22 contingencies: (i) In 6 contingencies, CART did not reach the same test accuracy as OCT*(S) across all studied depths. (ii) In the remaining 16 DTs, an average DT depth of the equivalent CART was estimated at 3.56, consequently, the average reduction in DT depth from CART to OCT*(S) was 1.56. To further provide information on the gain in

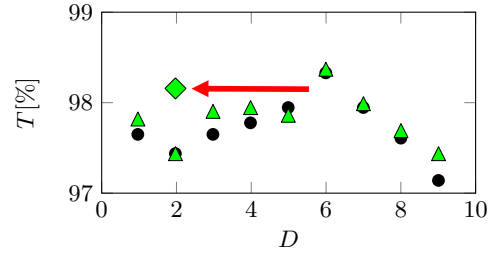


Fig. 6: Higher interpretability (smaller maximal DT depth D) of DTs at the same test accuracy T when using optimization-based DTs (OCT*(S) ◆) over CART DTs (●). GOT(S) (▲) with zero lookahead-level is also shown as a reference. Each data point corresponds to the average of 10 different DTs.

TABLE V: Estimation of the computational capability (offline) for the DSA of a large system.

Case study Approach	IEEE 68-bus system		French power system	
	CART	OCT*(S)	CART	OCT*(S)
Simulation time per observation	0.2 s		57 s [9]	
Required observations	2900	560	2900	560
Computational time of simulations	580 s	112 s	165 300 s	31 920 s
Time to learn DT with $D = 2$	1 s	306 s	1 s	306 s
Total	581 s	418 s	165 300 s	32 226 s

interpretability, the average reduction in terminal nodes was computed across all 16 contingencies to be 12.5. In other words, whereas OCT*(S) partitioned the feature space into 4 regions, CART required 16.5 to achieve the same testing accuracy.

G. Computational capability for large systems

To roughly estimate the computational capability of the full offline workflow for large power systems, the results from Sections IV-B, IV-E were projected to the French power system by using results from [9]. It is important to note that simulation time scales non-linearly with system size. Thus, to sample one observation is computationally much more expensive in larger systems. For example, the average time to simulate a single contingency was 0.2 s in the IEEE 68-bus system and 57 s [9] in the French power system. Note that no other data than this average take was taken from the French power system.

Here, we explore the balance in the computational time of (i) creating a training database and (ii) learning a DT. One can either use a large training database and a simple DT learning approach, such as CART, or a smaller training database and an advanced DT learning approach, such as OCT*(S). In terms of (i), Section IV-E showed that OCT*(S) result in savings in the required training observations (2900 versus 560), consequently, reducing the number of dynamic simulations required, which is the most severe computational bottleneck of DSA platforms. However, in terms of (ii), Section IV-B showed that OCT*(S) require higher computational times than CART (306 s versus 1 s). To study this balance in terms of computational time of (i) and (ii), it was assumed that the computational times to learn the DT with CART and OCT*(S) will be similar when scaling to the French power system. This

assumption is based that in both systems, the same subset of features would be pre-selected with a user-specified size. All assumptions and results of this estimation are stated in Table V. In the IEEE 68-bus system, OCT*(S) resulted in overall computational savings of 30 % over CART. When projecting the assumptions from the IEEE 68-bus system to the French power system, the total computational savings were as large as 80 % due to the higher simulation time per observation. This rough estimation illustrates that the larger the system, the greater the computational benefits of using OCT*(S) over CART.

H. Discussion

The optimization-based DTs have shown promising results. In short, although CART could also be used to process DSA, optimization-based DTs improve over CART DTs the trade-off between interpretability and predictive performance. This improvement may translate to reducing significantly the amount of data that is required to learn a DT. In more detail, we have shown that (i) our proposed OCT*(S) approach is 40 % faster than OCT(S) while keeping the same predictive accuracy, (ii) the use of optimization allows for increased flexibility in the DT splitting scheme, (iii) the predictive performance of both GOT(S) and OCT*(S) results in improving the test accuracy by an average of 1 % compared to CART, (iv) the number of required observations can be reduced by up to 80 %, (v) the interpretability is increasing as the DT depth can be reduced by more than 1.5, and, finally, (vi) the overall computational load of DSA platforms for large power systems can be significantly reduced by 80 %. This last result enables operators to (re)evaluate their distribution of computational resources along the offline workflow. This is a fundamental step forward to deploy security rules in large systems. However, optimization-based DTs have also limitations.

The key limitation lays in the scalability for large systems when using the proposed approaches without considering any other machine learning concepts in combination. Note that the proposed approaches require to be combined with other approaches into a full machine learning workflow. This limitation corresponds to the computational complexity which grows nonlinearly with DT depth, number of observations and number of features. In terms of DT depth, this is not an issue as small DTs are more interpretable, therefore, they are preferred. Moreover, both approaches OCT*(S) and GOT(S) are supposed to be used for different DT depths. Whereas OCT*(S) can be used for very small DT depths, GOT(S) can be used for deeper DTs. Increasing system size/complexity leads to a non-linear increase of observations/dynamic simulations required. However, the proposed approaches require fewer observations than other approaches. For this matter, we suggest combining the proposed OCT*(S) approach with a high-quality training database. Such a database can be obtained by using importance sampling techniques to maximize the information of each observation. Finally, to handle a large number of features of the original, large system problem, we suggest combining the optimization-based DTs with pre-selecting important variables; for example, we have used a ranking based on gini importance.

V. CONCLUSION

The need for interpretable and optimal security rules have been presented, showing that current approaches lead to complex security rules. In response, we proposed two optimization-based tree learning approaches: learning an optimal tree for small tree depths and a greedy optimization-based tree for intermediate tree depths. Through case studies on the IEEE 68-bus system, the proposed approaches were shown to improve the computational performance of a current optimization-based tree learning approach. Further, the proposed optimization-based approaches achieve superior capabilities in capturing the underlying characteristics of the data: resulting in higher predictive performances, higher interpretability of the security rules and less required training data, which can significantly reduce the overall offline computational load of dynamic security assessment platforms. This work enables, for the first time, to learn optimal security rules in order to increase the trust of operators to use security rules in the dynamic security assessment. In the future, this work would benefit from combining it with more advanced feature selection methods, with importance sampling and with a robust framework for changes in the probability distributions between training and testing, thus, that a full machine-learning workflow can be built in order to learn high-performance security rules.

ACKNOWLEDGMENT

This work was supported by a studentship funded by the Engineering and Physical Sciences Research Council. We are thankful to Nicolas Omont, Benjamin Donnot, Jean Maeght, Antoine Marot from Réseau de Transport d'Electricite who provided expertise that greatly assisted this research. We also thank Georgios Anagnostou whose dataset we used and Michael Evans for proofreading this manuscript.

REFERENCES

- [1] P. Panciatici, G. Bareux, and L. Wehenkel, "Operating in the fog: Security management under uncertainty," *IEEE Power and Energy Magazine*, vol. 10, no. 5, pp. 40–49, 2012.
- [2] L. A. Wehenkel, *Automatic Learning Techniques in Power Systems*. Kluwer Academic Publishers, 1998.
- [3] R. Diao, V. Vittal, and N. Logic, "Design of a real-time security assessment tool for situational awareness enhancement in modern power systems," *IEEE Transactions on Power Systems*, vol. 25, no. 2, pp. 957–965, 2010.
- [4] J. L. Cremer, I. Konstantelos, S. H. Tindemans, and G. Strbac, "Sample-derived disjunctive rules for secure power system operation," in *2018 International Conference on Probabilistic Methods Applied to Power Systems (PMAPS)*. IEEE, 2018, Conference Proceedings.
- [5] K. Sun, S. Likhate, V. Vittal, V. S. Kolluri, and S. Mandal, "An online dynamic security assessment scheme using phasor measurements and decision trees," *IEEE Transactions on Power Systems*, vol. 22, no. 4, pp. 1935–1943, 2007.
- [6] V. Krishnan, J. D. McCalley, S. Henry, and S. Issad, "Efficient database generation for decision tree based power system security assessment," *IEEE Transactions on Power Systems*, vol. 26, no. 4, pp. 2319–2327, 2011.
- [7] C. Liu, K. Sun, Z. H. Rather, Z. Chen, C. L. Bak, P. Thogersen, and P. Lund, "A systematic approach for dynamic security assessment and the corresponding preventive control scheme based on decision trees," in *2014 IEEE PES General Meeting — Conference & Exposition*, Conference Proceedings, pp. 1–1.

BIBLIOGRAPHIES

- [8] M. H. Vasconcelos, L. M. Carvalho, J. Meirinhos, N. Omont, P. Gambier-Morel, G. Jamgotchian, D. Cirio, E. Ciapessoni, A. Pitto, I. Konstantelos, G. Strbac, M. Ferraro, and C. Biasuzzi, "Online security assessment with load and renewable generation uncertainty: The itesla project approach," in *2016 International Conference on Probabilistic Methods Applied to Power Systems (PMAPS)*, Conference Proceedings, pp. 1–8.
- [9] I. Konstantelos, G. Jamgotchian, S. Tindemans, P. Duchesne, S. Cole, C. Merckx, G. Strbac, and P. Panciatici, "Implementation of a massively parallel dynamic security assessment platform for large-scale grids," *IEEE Transactions on Smart Grid*, vol. PP, no. 99, pp. 1–1, 2016.
- [10] H. Sun, F. Zhao, H. Wang, K. Wang, W. Jiang, Q. Guo, B. Zhang, and L. Wehenkel, "Automatic learning of fine operating rules for online power system security control," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 27, no. 8, pp. 1708–1719, 2016.
- [11] F. Doshi-Velez and B. Kim, "Towards a rigorous science of interpretable machine learning," *arXiv preprint arXiv:1702.08608*, 2017.
- [12] M. Sun, I. Konstantelos, and G. Strbac, "A deep learning-based feature extraction framework for system security assessment," *IEEE Transactions on Smart Grid*, pp. 1–1, 2018.
- [13] S. R. Samantara, I. Kamwa, and G. Joos, "Ensemble decision trees for phasor measurement unit-based wide-area security assessment in the operations time frame," *IET Generation, Transmission & Distribution*, vol. 4, no. 12, pp. 1334–1348, 2010.
- [14] J. Cremer, I. Konstantelos, S. Tindemans, and G. Strbac, "Data-driven power system operation: Exploring the balance between cost and risk," *IEEE Transactions on Power Systems*, 2018.
- [15] C. Liu, F. Tang, and C. Leth Bak, "An accurate online dynamic security assessment scheme based on random forest," *Energies*, vol. 11, no. 7, p. 1914, 2018.
- [16] I. Houben, L. Wehenkel, and M. Pavella, "Coupling of k-nn with decision trees for power system transient stability assessment," in *Control Applications, 1995., Proceedings of the 4th IEEE Conference on*. IEEE, Conference Proceedings, pp. 825–832.
- [17] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, "Classification and regression trees," *Wadsworth & Brooks Monterey, CA*, 1984.
- [18] J. Mingers, "An empirical comparison of pruning methods for decision tree induction," *Machine learning*, vol. 4, no. 2, pp. 227–243, 1989.
- [19] S. Esmeir and S. Markovitch, "Anytime learning of decision trees," *Journal of Machine Learning Research*, vol. 8, no. May, pp. 891–933, 2007.
- [20] S. W. Norton, "Generating better decision trees," in *IJCAI*, vol. 89, 1989, Conference Proceedings, pp. 800–805.
- [21] S. Murthy and S. Salzberg, "Lookahead and pathology in decision tree induction," in *IJCAI*. Citeseer, 1995, Conference Proceedings, pp. 1025–1033.
- [22] M. Norouzi, M. Collins, M. A. Johnson, D. J. Fleet, and P. Kohli, "Efficient non-greedy optimization of decision trees," in *Advances in Neural Information Processing Systems*, 2015, Conference Proceedings, pp. 1729–1737.
- [23] D. Bertsimas and J. Dunn, "Optimal classification trees," *Machine Learning*, vol. 106, no. 7, pp. 1039–1082, 2017.
- [24] E. Balas, "Disjunctive programming and a hierarchy of relaxations for discrete optimization problems," *SIAM Journal on Algebraic Discrete Methods*, vol. 6, no. 3, pp. 466–486, 1985.
- [25] F. Trespacios and I. E. Grossmann, "Improved big-m reformulation for generalized disjunctive programs," *Computers & Chemical Engineering*, vol. 76, pp. 98–103, 2015.
- [26] B. Pal and B. Chaudhuri, *Robust control in power systems*. Springer Science & Business Media, 2006.
- [27] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical programming*, vol. 106, no. 1, pp. 25–57, 2006.
- [28] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, and V. Dubourg, "Scikit-learn: Machine learning in python," *Journal of Machine Learning Research*, vol. 12, no. Oct, pp. 2825–2830, 2011.
- [29] W. E. Hart, C. D. Laird, J.-P. Watson, D. L. Woodruff, G. A. Hackebeil, B. L. Nicholson, and J. D. Siirola, *Pyomo-optimization modeling in python*. Springer Science & Business Media, 2017, vol. 67.

Jochen L. Cremer (GSM17) received the B.Sc. degree in mechanical engineering in 2014, the B.Sc. degree in electrical engineering in 2016 and the M.Sc. degree in chemical engineering in 2016, all from the RWTH Aachen University, Germany. Currently he works towards the Ph.D. degree in the Control and Power Research Group, Imperial College London, U.K. His research interests include machine learning and mathematical programming applied to the operation and planning of power systems.

Ioannis Konstantelos (M12) received the MEng degree in Electrical and Electronic Engineering from Imperial College London in 2007. He obtained his PhD from the same university in 2013 in the field of electrical energy systems. His research interests include mathematical programming and machine learning techniques applied to the planning and operation of energy systems.

Goran Strbac (M95) is a Professor of electrical energy systems at Imperial College, London, UK. His current research interests include electricity generation, transmission and distribution operation, planning and pricing, and integration of renewable and distributed generation in electricity systems.