Fast and Explainable Warm-start Point Learning for AC Optimal Power Flow Using Decision Tree

Yuji Cao[a], Huan Zhao[c,*], Gaoqi Liang[a,b], Junhua Zhao[a,b,**], Huanxin Liao[a,b] and Chao Yang[b]

[a]organization=Shenzhen Institute of Artificial Intelligence and Robotics for Society (AIRS), city=Shenzhen, postcode=518129, country=China
[b]organization=School of Science and Engineering, The Chinese University of Hong Kong, city=Shenzhen, postcode=518172, country=China
[c]organization=School of Electrical and Electronic Engineering, Nanyang Technological University, postcode=639798, country=Singapore

ARTICLE INFO

Keywords:
Alternating Current Optimal Power Flow
Warm-start point
Decision tree
Interpretable model

ABSTRACT

The quality of starting point greatly influences the result and convergence efficiency of the optimization algorithm, especially for the non-convex and constrained Alternating Current Optimal Power Flow problem. Generally, speed and accuracy are the two main evaluation metrics for generating starting points. The data-driven methods learn the starting point through historical data and show good performance. However, most methods utilize "black-box" models, which lack interpretability. Therefore, this paper proposes a fast and explainable warm-start point learning method based on the multi-target binary decision tree with a post-pruning module. The calculated warm-start points can accelerate the solving process and the model inference time is extremely short. The post-pruning module is applied to fit different power system scenarios fairly and alleviate the overfitting problem by pruning the completely grown tree. Also, a set of detailed decision rules for selecting warm-start points are generated after the learning process. The generated rules assist the power system operators in identifying important loads and thereby provide the model interpretability. The experiment shows that the proposed framework can reduce the solving times for the Alternating Current Optimal Power Flow solvers with an extremely short calculation time for the explainable warm-start point.

1. Introduction

In a power system, the power generation is dispatched by solving the Alternating Current Optimal Power Flow (AC-OPF) problem. Caused by the high uncertain fluctuations in loads and renewable energy, the AC-OPF problems need to be solved more frequently and accurately to ensure the stability and safety of the system [1]. However, due to the non-convexity nature of the AC-OPF problem, it is a challenging problem to find the optimal solutions within a demanding time [2].

Normally, a warm-start point ensures convergence and significantly reduces iteration times [3]. Based on the warm-start points, the feasibility of the solution is ensured and thus is more reliable than the direct approach. The grid operator generally uses a flat start or the DC-OPF solution as the starting point of the optimization process [4]. However, the flat start points may not guarantee convergence and the DC-OPF solution still involves an iterative solving process that requires extra computational time [5]. To address the problem caused by poorly chosen initial points, reference [6], [7] and [8] use a continuation-based method. Nevertheless, the slow calculation is not suitable with highly stochastic intermittent resources nowadays [9].

As a large amount of data has been generated in the smart grid [10], various data-driven approaches for the AC-OPF problem have been proposed, which can be classified into direct approaches and indirect approaches. The direct approach predicts the solutions directly without utilizing the optimization algorithms. The majority uses deep neural networks (DNN) to predict the generation directly. In [11], the author uses the deep reinforcement learning method for fast AC-OPF solutions. However, the feasibility of the solution and the constraints are hard to be satisfied. To address the feasibility issue, methods in [12] integrate the constraints into the reward function to penalize the violation. The indirect approach accelerates the solving process by providing warm-start points to initialize the physics-based solver (e.g., MATPOWER Interior Point Solver (MIPS)). Existing methods include random forest [13] and artificial neural networks in [14], [15], and [16]. Nevertheless, the former is an ensemble learning method that uses different models to take the average output, which is slow in computation due to its model complexity [17]. The latter is a black-box model composed of different interconnected nodes, which lacks interpretability, and sometimes is not a reliable choice for grid operators [18].

Interpretability is critical in a complicated physical system for security concerns [19]. One crucial security aspect for operators is the overall awareness of the methods applied to the system, such as how the methods work, the worst performance, etc. Previously, interpretability existed inherently in the physical optimal power flow model [20]. Nowadays, various machine learning and deep learning models have been proposed to solve the AC-OPF problem such as methods in [21] and [22]. However, most of them are black-box, and thus operators cannot grasp the decision making process of the model. Therefore, models without interpretability and transparency are a major risk to the power system, especially under safety-critical scenarios. To address this

*Corresponding author
**The author have the same contribution as the corresponding author.
Email addresses: yujicao@link.cuhk.edu.cn (Y. Cao); huan.zhao@ntu.edu.sg (H. Zhao); gaoqi.liang@outlook.com (G. Liang); zhaojunhua@cuhk.edu.cn (J. Zhao); huanxinliao@link.cuhk.edu.cn (H. Liao); yangchao@cuhk.edu.cn (C. Yang)
ORCID(s):

problem, different interpretable models have been proposed. In [23], a XGBoost model is adopted to evaluate the feature importance and thus provide certain model interpretability in the data selection part. In [24], the method constructs oblique classification trees to learn the optimal strategy from mixed-integer quadratic problems and the tree helps provide interpretability. Both employ tree-based methods to provide interpretability. Indeed, compared with other black-box machine learning methods, the tree-based model has a clear underlying structure and set of rules for interpretation [25].

To meet both the needs for fast computation and interpretability, this paper proposes a warm-start point learning method for AC-OPF based on the multi-target decision tree with a post-pruning module. The contributions are summarized as follows. First, a multi-target binary decision tree-based model is proposed to provide warm-start points for the AC-OPF problem. The calculated warm-start points accelerate the AC-OPF solving process significantly and the model inference time is extremely short (on the microsecond timescale). Second, to fit different power system scenarios fairly and alleviate the overfitting problem, a post-pruning method is adopted. Third, a set of decision rules can be extracted from the trained models to explain the power system considerations behind the calculated warm-start points. The extracted results help the power system operators identify important loads and thus provide model interpretability.

The remaining sections of this paper are organized as follows. Section II presents the background and notation of the AC-OPF problem and decision tree model. Section III introduces the proposed framework where the decision tree model is applied to predict the warm-start point with interpretability. Then the proposed framework is validated on three IEEE cases, and the results are presented in Section IV. The conclusion is provided in the last section.

2. Background and Notation

2.1. AC Optimal Power Flow

AC-OPF problem aims to minimize the operation costs by finding the optimal power output under the given loads while satisfying physical and security constraints [26]. The standard AC-OPF problem can be formulated as below:

$$\min \sum_{i\in\mathcal{N}_G} C(P_{gi}) \tag{1}$$

$$\begin{aligned} \text{s.t.} \quad & P_{gi} - P_{di} = V_i \sum_{j\in\mathcal{N}} V_j(g_{ij}\cos\theta_{ij} + b_{ij}\sin\theta_{ij}), i \in \mathcal{N} \\ & Q_{gi} - Q_{di} = V_i \sum_{j\in\mathcal{N}} V_j(g_{ij}\sin\theta_{ij} - b_{ij}\cos\theta_{ij}), i \in \mathcal{N} \end{aligned} \tag{2}$$

$$P_{gi}^{min} \le P_{gi} \le P_{gi}^{max}, i \in \mathcal{N}_G \tag{3}$$

$$Q_{gi}^{min} \le Q_{gi} \le Q_{gi}^{max}, i \in \mathcal{N}_G \tag{4}$$

$$V_i^{min} \le V_i \le V_i^{max}, i \in \mathcal{N} \tag{5}$$

$$P_{ij}^2 + Q_{ij}^2 \le (S_{ij}^{max})^2, (i,j) \in \xi \tag{6}$$

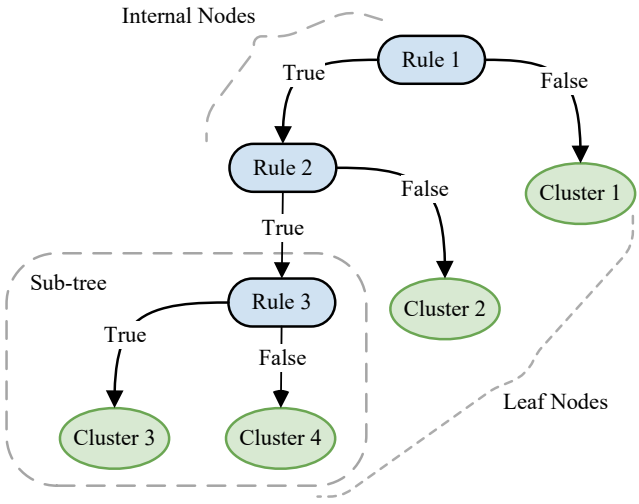

Figure 1: Decision tree structure

$$\theta_{ij}^{min} \le \theta_{ij} \le \theta_{ij}^{max} \tag{7}$$

where $\mathcal{N}$, $\mathcal{N}_G$ and ξ represent the set of all buses, generation buses, and lines, respectively; V_i and θ_i are the voltage magnitude and angle; $\theta_{ij} = \theta_i - \theta_j$; g_{ij} and b_{ij} are the conductance, susceptance of branch (i,j), respectively; P_{gi}, Q_{gi}, P_{di}, Q_{di} denote active power, reactive power, active load, and reactive load, respectively; S_{ij}^{max} is the branch flow limit of the branch (i,j). For a variable x, x^{min} and x^{max} represent its lower and upper bound, respectively. The constraints include the power-flow balance equations in (2), the active and reactive power limits in (3) and (4), the voltage magnitude limits in (5) and branch flow limits in (6), and voltage angle limits in (7).

2.2. Decision Tree

The decision tree is a non-parametric supervised learning method used for regression and classification [27]. The goal is to build a model that predicts the value of a target variable based on a series of decision rules inferred from the data features. The following introduces the structure, decision rules, and the learning process.

2.2.1. Decision Tree Structure

The decision tree model is a tree-like structure that classifies the data points, as shown in Fig. 1. It consists of nodes and directed edges. The nodes include internal nodes and leaf nodes, where an internal node represents a feature of the input data, and a leaf node represents a class.

The classification process of a data point contains the following steps:

1. From the root node, the tree performs a test on a feature of the data point.
2. According to the test result, the node distributes the data point to one of its sub-nodes. The sub-node corresponds to one of the values of the feature.
3. The tree recursively tests and distributes the data point until the data point falls into a leaf node.

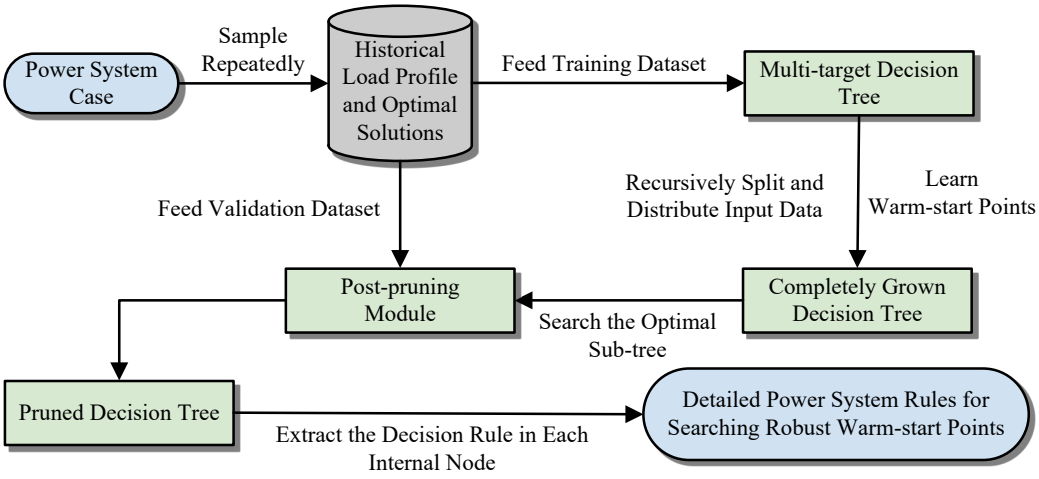

Figure 2: Multi-target binary decision tree framework with interpretability

4. The data point is classified as the class of the leaf node.

2.2.2. Explanation of Decision Tree

The decision tree can be viewed as a set of if-then rules according to the path from the root node to the leaf nodes. The feature of each internal node corresponds to one rule. The branch from the internal node is the outcome of the rule. The class of the leaf node is the conclusion of a combination of rules. Note that the if-then rules should be mutually exclusive and complete, which means each data point is covered by one rule and can only be covered by one rule. By following the decision path, the input data space is recursively divided into sub-spaces according to the rules, and the leaf node is the result of the division.

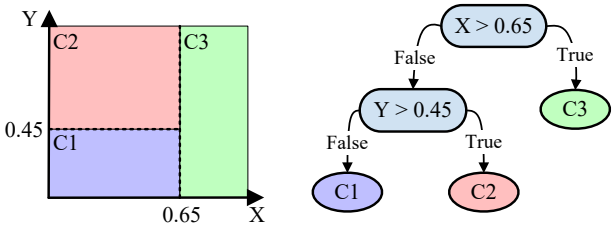

Figure 3: Exemplary illustration of a decision tree in two-dimensional feature space

An example of the division is shown in Fig. 3. According to the value of X and Y, the trained decision tree divides the input data space into three regions and classifies each region's data points accordingly.

3. Framework

3.1. Warm-start Point Learning and Explanation Framework

Initial points effect the convergence and calculational time in the AC-OPF problem. A poorly chosen initial point may result in failure of convergence. A warm-start point provides an initial point that greatly reduces the computational time and increases the stability of the solving process. In scenarios that require frequent AC-OPF calculations such as online market clearing (every 5 minutes in New England), a fast and efficient warm-start calculation saves considerable time for the power grid operations. Generally, a warm-start point is obtained by solving a simplified optimization problem or using the related historical data. In our proposed framework, warm-start points of an AC-OPF problem are learned using historical optimal solutions and the quality of historical optimal solutions determines the convergence and calculational time of the learned warm-start points.

In the previous literature regarding to warm-start points, voltage magnitude and active power generation are selected as target variables [13]. This paper employs the same setting. Since potential correlations exist between the two variables, to better preserve the correlations and provide a warm-start point, a multi-target binary decision tree is proposed to predict both of the target variables simultaneously. The decision tree learns the warm-start points using the historical load profiles and their corresponding optimal voltage magnitude and active power generation. However, the learned decision tree is prone to be overfitting in some network scenarios other than learning the general warm-start patterns in the network. To address this problem, a post-pruning module is applied to alleviate the risks by pruning the tree and thus restricting its complexity. Last, given the optimal pruned decision tree, a set of decision rules can be extracted among the internal nodes. By extracting the rules, the decision path is clear and transparent for operators. Also, important load buses can be filtered out by considering the order of selected load buses. Therefore, the model is interpretable and explainable.

The whole proposed framework is demonstrated in Fig. 2. In the data preparation part, the AC optimal power flow is repeatedly solved with different load profiles to create a dataset, where the historical load demands $[P_d, Q_d]$ of each

bus and the corresponding voltage magnitudes and active power generation $[P_g, V_g]$ are stored as pairs. Given the training dataset, a decision tree learns the decision rules and recursively split its internal nodes to distribute the input data and calculate the warm-start points. After the learning process, a completely grown decision tree is constructed. Then a post-pruning module is applied to the tree and the optimal sub-tree is selected. After that, by extracting the decision rules in each internal node, a set of detailed power system rules for search warm-start can be obtained for future use.

3.2. Multi-target Binary Decision Tree Model

Generally, a regression decision tree learns the optimal prediction value by finding the optimal splitting feature and value of all nodes, which minimizes the overall mean squared prediction error in leaf nodes. For the AC optimal power flow problem, a learned binary decision tree can find the warm-start point according to the leaf node to which it belongs. This searching process only needs several comparisons. Therefore, compared with complex multiplications of large matrices in deep learning, the calculation is much more fast and thus a warm-start point can be provided within a short time.

In this paper, the binary decision tree splits the input load profile data according to the variance of their corresponding optimal solutions. After the splitting, load profiles with similar optimal solutions, e.g., similar network scenarios, are grouped together. This process is repeated until the predefined termination condition is reached. Based on the resulting groups, the decision tree learns their warm-start points using the center of their optimal solutions. The details are as follows.

Suppose X is the input variable and Y is the continuous output variable for this model, they are defined as below:

$$X = [P_{d1}, \dots, P_{dN}, Q_{d1}, \dots, Q_{dN}]$$
$$Y = [P_{g1}, \dots, P_{gN_G}, V_1, \dots, V_N]$$

where N is the number of buses and N_G is the number of generation buses.

For a grown decision tree with M leaf nodes, the output and prediction loss are:

$$f(x) = \sum_{m=1}^{M} c_m I(x \in R_m) \tag{8}$$

$$L = \sum_{m=1}^{M} (y_i - f(x_i))^2 \tag{9}$$

where I is the indicator function and c_m is the output value of leaf node m. At each leaf node, assuming there are N_m samples, the optimal output value c_m is represented by (10). It is the average output of samples in the leaf node obtained by minimizing the mean squared prediction loss of the leaf node.

$$\hat{c}_m = \frac{1}{N_m} \sum_{x_i \in R_m} y_i \tag{10}$$

In the learning process, at each internal node, the decision tree selects a feature $x^{(j)}$ and its corresponding value s as the splitting feature and splitting points, respectively. According to the splitting feature and value, the node is divided into two sub-nodes, which are shown below:

$$R_1 = \{x | x^{(j)} \le s\} \tag{11}$$

$$R_2 = \{x | x^{(j)} > s\} \tag{12}$$

To find the optimal splitting feature j and its splitting value s, the decision tree adopts the heuristic method to solve the optimization problem below:

$$\min_{j,s} \left[\min_{C_{R_1}} L_{R_1(j,s)} + \min_{C_{R_2}} L_{R_2(j,s)} \right] \tag{13}$$

$$L_R = \sum_{x_i \in R} (y_i - C_R)^2 \tag{14}$$

$$\hat{c}_R = \frac{1}{N_{r(j,s)}} \sum_{x_i \in R(j,s)} y_i \tag{15}$$

where $\hat{c}_R$ is the predicted output in each sub-node since it is the optimal solution to minimize L_R.

Algorithm 1 Multi-target Binary Decision Tree Learning

Input: X, Y, tree T with only one root node
Output: A completely grown decision tree
1: *Initialization*: Initially all input data are stored in the root node
2: Start from the root node, partition the input data using the following recursive procedure
3: **for** nodes in leaf nodes **do**
4: **if** tree depth < predefined maximum depth and number of samples > predefined minimum node size **then**
5: Solve (13) to find the optimal splitting feature j^* and the optimal splitting value s^*
6: Split node into two sub-nodes
7: Use selected optimal (j^*, s^*) pair to distribute input data into sub-nodes by (11) and (12)
8: Calculate the optimal predicted output of each sub-node by (15)
9: **end if**
10: **end for**
11: **Return** T

The learning algorithm for a multi-target binary decision tree model is shown in Algorithm 1. In the beginning, all input data are in the root node. Then from the root node, the decision tree distributes the data with steps 5-8. Suppose the predefined termination conditions (e.g. maximum depth or minimum node size) are not satisfied in the leaf node. In that case, it will heuristically find the optimal splitting feature and value to split the leaf node and distribute the input into its sub-nodes until the predefined conditions are fulfilled.

3.3. Post-pruning Module

During the learning process of the decision tree, the tree tries to minimize the prediction error by splitting the

internal nodes as much as possible. This results in a large and complex tree with plenty of branches and nodes, and it is likely to fit closely to the existing data but fails to capture the important patterns of unknown network scenarios reliably. However, deciding the termination condition is hard, since it is unrealistic to be aware of whether a single extra tree node will reduce the error for future scenarios. Therefore, a cost complexity post-pruning module is applied to the completely generated decision tree. The module evaluates each sub-tree on the validation data by leveraging a loss function with a penalty added to the model complexity. Thus, a trade-off between training loss and model complexity is achieved to optimize the performance.

The post-pruning module recursively prunes some sub-trees from the bottom of the completely grown decision tree T_0 to generate a sub-tree sequence $[T_0, T_1, \dots, T_n]$. An optimal sub-tree is selected according to the performance of the independent validation set.

In the pruning process, for a sub-tree T, to measure the overall loss and the effectiveness of pruning, two metric functions are defined below, respectively:

$$C_\alpha(T) = C(T) + \alpha|T| \tag{16}$$

$$g(t) = \frac{C(t) - C(T_t)}{|T_t| - 1} \tag{17}$$

where $|T|$ is the number of leaf nodes, $C(T)$ is the prediction loss of T, T_t is the sub-tree with root node t, and $\alpha \geq 0$ is the penalty parameter to balance the trade-off between the model complexity and the data fitting. $g(t)$ measures the decrease of overall loss after pruning the sub-tree of an internal node t.

The algorithm for the post-pruning process is shown in Algorithm 2. Given a completely grown decision tree T_0, in the beginning, the algorithm initializes k to 0, T to T_0, and α to positive infinity. Then for each internal node t in T, the algorithm computes the metrics in (12) and (13) and set α to the minimum value between $g(t)$ and itself. Then the sub-tree of internal nodes whose $g(t)$ equals α are pruned and k, α_k, and T_k is reset until the current tree T_k contains only a root node and two leaf nodes. After obtaining the sub-tree sequence $T_0, T_1, \dots, T_n$, the optimal sub-tree T_α is selected with the cross-validation methods.

Algorithm 2 Post-pruning process

Input: Validation set, a complete grown decision tree T
Output: An optimal pruned sub-tree T_α

Initialization: Initialize $k = 0$, $T = T_0$
Set $\alpha = +\infty$
for each internal node t in T **do**
  Calculate $C(T_t)$, $|T_t|$, $g(t)$, $\alpha = \min(\alpha, g(t))$
  if $g(t) == \alpha$ **then**
    Prune internal node t
  end if
  Set $k = k + 1$, $\alpha_k = \alpha$ and $T_k = T$
  if T_k contains only a root node and two leaf nodes **then**
    Set $T_k = T_n$
  else
    Go back to Step 2
  end if
end for
Use the cross-validation method to select the optimal sub-tree T_α from sequence $T_0, T_1, \dots, T_n$
return T_α

3.4. Decision Rules Extraction

Each internal node of the decision tree contains a decision rule. Benefitting from this characteristic, the decision tree is interpretable and reliable compared with other machine learning models. Additionally, by extracting the decision rules, different information, such as important load buses, similarity within each group of scenarios, and dissimilarity between different scenarios, can be filtered out to provide valuable insights for grid operators.

For the AC-OPF problem, a decision rule is a comparison between the load value and a learned threshold value of a specific bus. From the root node to each leaf node, each important load bus is selected greedily for splitting. In this process, the order of selected buses represents their importance in reducing the prediction error. For instance, the selected bus in the root node reduces the prediction error the most and contains the most information, and thus it is the most important load bus under the model consideration. From the top to the bottom of a decision tree, the importance of selected buses is ordered descendingly.

The algorithm to extract detailed decision rules for generating a warm-start point in a power system is shown in Algorithm 3. Given a pruned decision tree, for each internal node inside the tree, the load bus and threshold value are found. As a pair, the comparison between them is constructed as an if-statement. The splitting result is constructed as a then-else statement. Last, the leaf nodes are constructed as the result of rules.

Algorithm 3 Decision Rules Extraction

Input: A decision tree T
Output: A set of decision rules

for each internal node t in T **do**
  Extract the splitting feature j and threshold s
  Find load bus i of the splitting feature j
  for each load bus i and threshold s **do**
    Construct an if-statement based on the comparison of the load in bus i and threshold s
    Construct a then-else statement based on the comparison result
  end for
end for
The leaf nodes are the result of rules
return A set of decision rules

4. Case Study

4.1. Setup

The proposed framework is evaluated numerically on IEEE 57-bus, 118-bus, and 300-bus networks. All experiments are performed on a MacBook Air with an M1 chip and 16GB RAM. The training and test datasets are generated by MATPOWER [28] by solving the AC-OPF problem with different starting points.

The starting points were generated as follows. The load demands were uniformly sampled within $[80\%, 120\%]$ of the default load profile for each IEEE case. Besides, in each case, a part of randomly selected load demands are set to 0. Then load demands were solved repeatedly, and the feasible solutions were collected in the dataset. In total, 20000 samples were collected for each test case. Then the dataset was split into the training set and test set by 80/20. To evaluate the effectiveness of different warm-start methods, a group of interpretable data-driven methods is used, including: decision tree, random forest, and linear regression. The neural network based methods are not included in the comparison due to the lack of interpretability.. The random forest model is trained with the optimal configuration in [13] and the hyperparameters are listed in Table 1. After training, the predicted warm-start output is solved by the MATPOWER's MIPS (MATPOWER Interior Point Method) with the FMINCON solver.

Table 1
Random Forest Hyperparameters

HYPERPARAMETERS \TESTCASE	IEEE 57	IEEE 118	IEEE 300
Number of estimators	200	400	400
Max Depth	15	15	20
Minimum Samples Split	4	2	3

The completely grown decision tree is compared with the pre-pruning decision tree and post-pruning decision tree. The pre-pruning method pre-defines several hyperparameters including the maximum depth of the decision tree and the minimum percentage of samples in each split. Both of them are selected with a grid search. The maximum depth of the decision tree is selected from [6, 11] and the minimum percentage of samples in each split is selected from [0%, 2.5%, 5%, 7.5%].

4.2. Comparison of Different Warm-start Point Methods

Different warm-start point methods accelerate the optimization process by reducing the iterations. In this experiment, the iterations of the AC-OPF solver using the different warm-start point methods are compared. The proposed method is compared with the DC-OPF solution, random forest solution, and flat start (voltage magnitudes set to 1.0 p.u., angles and generation set to 0). The underlying assumption is that the iteration times of AC-OPF solver are proportional to the calculation time, since the calculation complexity in each iteration is roughly the same. Therefore, a fewer iteration times means a shorter calculation time.

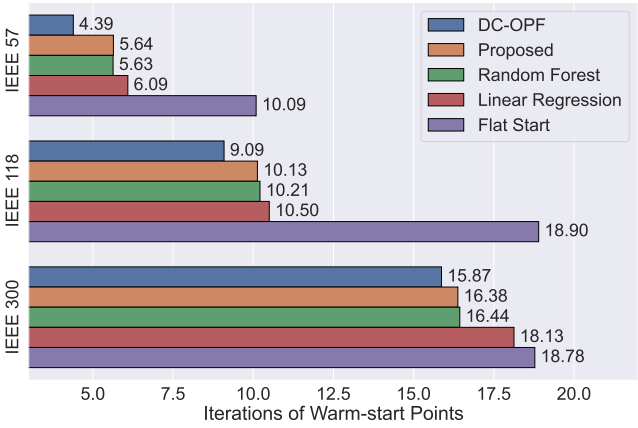

Figure 4: Comparison of different warm-start methods

Fig. 4. shows the average iteration result of 4000 sample points in the test datasets. In all three IEEE cases, the proposed framework achieves a comparable result. Compared with the flat start method, there is a 46.4% iteration reduction at most. The linear regression method performs the worst in machine learning-based methods, due to the non-linear and non-convex characteristics of the AC-OPF problem. Interestingly, the proposed method reduces more iterations in two out of three cases compared to the random forest with hundreds of estimators, indicating that the number of decision trees does not largely influence the model performance. As the number of buses increases, the scenarios become more complicated. For both DC-OPF solutions and machine learning methods, it is hard to find a start point that reduces the iterations significantly. Also, the performance differences between them decrease accordingly.

Table 2
Average Warm-Start Point Calculation Time (10^{-3} s)

METHOD \ TESTCASE	IEEE 57	IEEE 118	IEEE 300
Proposed Decision Tree	**0.02**	**0.02**	**0.02**
Random Forest	5.75	66.59	152.99
DC-OPF	315.20	345.20	382.00
Linear Regression	0.02	0.04	0.05

In the context of a growing number of distributed renewable energy nodes being integrated into the power grid, as the number of nodes increases, AC-OPF calculations become increasingly time-consuming in large-scale systems. Warm-start point methods can help accelerate the AC-OPF. However, though DC-OPF warm-start points are capable of time

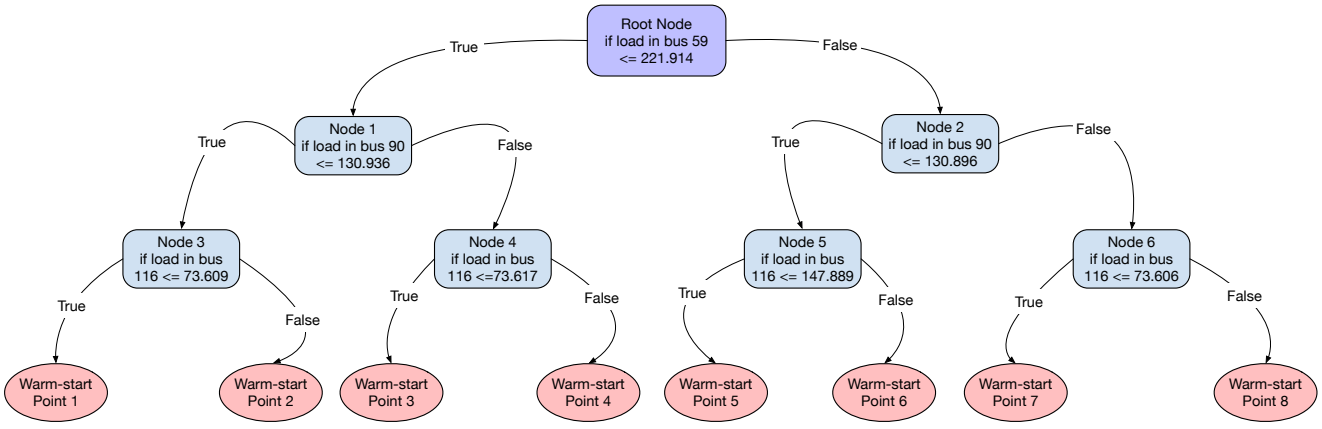

Figure 5: Tree-based decision rules extracted from post-pruned decision tree in IEEE 118 case

savings, the DC-OPF itself still requires a long calculation time in large-scale systems. For example, in the case2383wp, case9241pegase, and case13659pegase, the average calculation time are 1.14 s, 7.14 s, and 9.64 s, respectively. Compared to DC-OPF warm-start methods, data-driven methods can provide a warm-start point in a short time based on the historical data. Therefore, though the proposed decision tree only achieves a comparable result when compared to DC-OPF, calculation time of the proposed method has a huge acceleration. To measure the computational efficiency of different methods, we compared the average calculation time of 10 random samples in the test dataset. The results are shown in Table 2. On average, the proposed decision tree obtained one warm-start point around 2×10^{-5} s, while DC-OPF is around 3.47×10^{-1} s and the random forest is around 7.5×10^{-2} s. Linear regression has a similar calculation time to our proposed method, however, the start-points of linear regression have a marginal contribution to reducing the iterations. Also, compared with other methods, the average calculation time of our method does not change according to the complexity of the test case. The acceleration comes from the reason is that the proposed method only does several comparisons for all scenarios, while the random forest model involves hundreds of trees with large depths and DC-OPF performs large matrix multiplications in each iteration. Therefore, our method has a speedup of 17300 compared to DC-OPF and a speedup of 3700 compared to Random Forest. The result shows the huge acceleration in computational time of our proposed framework. Based on the results of iteration times and calculation time of warm-start points, the proposed method reduces more iteration times and has a short calculation time compared to other machine learning methods. For the DC-OPF warm-start method, though the iteration times are slightly better than the proposed method (1.23 iteration on average of three IEEE cases), it requires an extra large calculation time for a DC-OPF solution first compared to the proposed method. Therefore, the proposed obtained results help to shorten the overall calculation time.

4.3. Decision Rules Extraction

From a generated decision tree, detailed decision rules can be collected to provide information such as the importance of different load buses and interpretability for model decisions.

An example of pruned decision tree is illustrated in Fig. 5 and extracted tree-based decision rules is shown in Algorithm 4. At the root node, load in bus 55 is selected first to distribute the input data, which means it is the most important bus to find the warm-start point. Based on the load value in bus 55, different data points are distributed according to the evaluation result of the if-statement in the node. For instance, if the load value of a data point is smaller than 221.914, it falls into Node 1. Otherwise, it falls into Node 2. Afterward, load in bus 90 and load in bus 116 are selected to distinguish a data point. After the comparisons of the load values in these 3 buses, the tree divides the input data into 8 regions and each corresponds to a warm-start point. The extracted rules are helpful for system operators to determine important buses and also provide interpretability and reliability.

4.4. Comparison of Different Pruning Methods

In the framework, the post-pruning module is applied to a complete tree to alleviate the overfitting risks. To evaluate the effectiveness of the post-pruning module, we compare the iteration number of the decision tree without pruning, the decision tree with pre-pruning, and the decision tree with post-pruning.

The structures of different decision trees are shown in Table 3 and the iteration results of different decision trees (DT) are shown in Fig. 6. As shown in the table, while the complete decision tree splits one exact leaf node for each data point, all pruned trees remain a simpler structure with smaller depths and fewer leaf nodes. The effect of pruning is illustrated in the figure. The iterations of the pruned trees in different cases are reduced in all IEEE cases, indicating that the pruned sub-trees tend to add extra leaf nodes to minimize the training loss. By adding a penalty to the tree complexity and pruning these sub-trees, the resulting

Algorithm 4 Decision Rules Extraction Example

Input: A decision tree T
Output: A set of decision rules

```
if load in bus 59 ≤ 221.914 then
    if load in bus 90 ≤ 130.936 then
        if load in bus 116 ≤ 73.609 then
            Provide warm-start point 1
        else
            Provide warm-start point 2
        end if
    else
        if load in bus 116 ≤ 73.617 then
            Provide warm-start point 3
        else
            Provide warm-start point 4
        end if
    end if
else
    if load in bus 90 ≤ 130.896 then
        if load in bus 116 ≤ 147.889 then
            Provide warm-start point 5
        else
            Provide warm-start point 6
        end if
    else
        if load in bus 116 ≤ 73.606 then
            Provide warm-start point 7
        else
            Provide warm-start point 8
        end if
    end if
end if
```

pruned tree alleviates the risks of overfitting. Also, as the network scenarios become more complicated, the reduction of iteration times increases accordingly, which means the decision tree tends to over-fit more in complicated problems and the pruning module forces the tree to learn the general pattern of the network warm-start points by restricting its

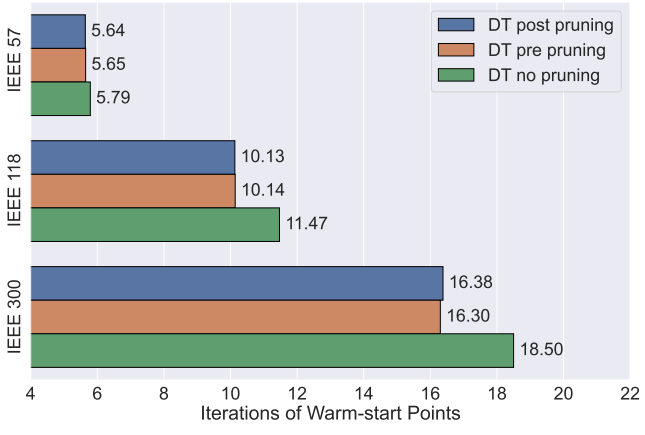

Figure 6: Comparison of different pruning methods

Table 3
Decision Tree Structure

MODELS\ TESTCASE	IEEE 57	IEEE 118	IEEE 300
No Pruning	16000 leaf nodes & depth = 26	16000 leaf nodes & depth = 24	16000 leaf nodes & depth = 21
Pre-pruning	255 leaf nodes & depth = 8	8 leaf nodes & depth = 3	8 leaf nodes & depth = 3
Post-pruning	355 leaf nodes & depth = 10	8 leaf nodes & depth = 3	34 leaf nodes & depth = 6

complexity. Last, pruned decision trees in IEEE 118 and IEEE 300 cases have a simpler structure than IEEE 57 cases, which means even though the network scenarios become more complex, the model is able to compute a warm-start point without increasing its complexity.

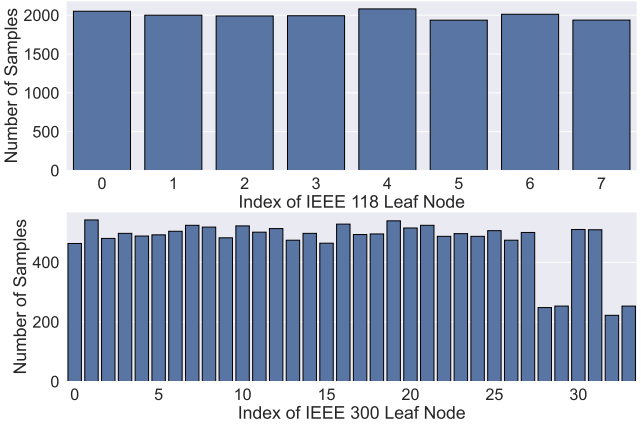

Figure 7: Number of leaf node samples in post-pruning decision tree

The number of leaf node samples in the post-pruning decision tree is shown in Fig. 7. With a simpler tree structure in IEEE 118-bus and IEEE 300-bus cases, the data samples fall evenly in most leaf nodes, indicating that the pruned decision tree is fair for each warm-start situation and has no clear bias for particular load profiles.

5. Conclusion

This paper proposes a decision-tree-based learning framework with the post-pruning method to provide warm-start points for accelerating the solving process of the Alternating Current Optimal Power Flow problem. The decision tree learns the warm-start points by recursively splitting the internal nodes based on the optimal dividing variables and values. The post-pruning module forces the proposed method to learn the general patterns of a given power system and thus alleviates the overfitting problem. Also,

the pruned decision tree remains a simpler structure, which provides a faster warm-start point calculation and clearer decision rules. Last, the extracted rules for a learned tree can reveal important load buses and thereby provide insights for power system operators. The test result verifies that 1) The proposed framework reduces the iterations of the Alternating Current Optimal Power Flow solver compared with the flat start method and achieves a slightly better result when compared with the random forest model; 2) Compared with the decision tree without pruning, the post-pruning module alleviates the overfitting problem and the model complexity is further reduced; 3) The proposed framework can calculate a warm-start point on a time scale of microseconds, compared the sub-second level of the random forest model and Direct Current Optimal Power Flow method.

Future work will explore reasonable ways to employ the proposed model on the larger-scale system to accelerate the AC-OPF with a fast and efficient warm start point. For example, using decentralized models with network decomposition methods with for better model scalability and reduce search space of the optimal solutions. Also, an important future direction is to properly incorporate the system topology information to further ensure the convergence and lower the iterations of the optimal power flow problem.

Acknowledgement

This work was supported by the National Natural Science Foundation of China, Grant/Award Numbers: 72171206, 42105145; Guangdong Province Natural Science Foundation (No. 2023A1515011438); the Shenzhen Key Lab of Crowd Intelligence Empowered Low-Carbon Energy Network (No. ZDSYS20220606100601002) and Shenzhen Institute of Artificial Intelligence and Robotics for Society.

References

[1] A. Abedi, M. R. Hesamzadeh, F. Romerio, Adaptive robust vulnerability analysis of power systems under uncertainty: A multilevel opf-based optimization approach, Int. J. Electr. Power Energy Syst. 134 (2022) 107432.

[2] D. K. Molzahn, Identifying and characterizing non-convexities in feasible spaces of optimal power flow problems, IEEE Transactions on Circuits and Systems II: Express Briefs 65 (2018) 672–676.

[3] A. Venzke, S. Chatzivasileiadis, D. K. Molzahn, Inexact convex relaxations for ac optimal power flow: Towards ac feasibility, Electric Power Systems Research 187 (2020) 106480.

[4] F. D. Freitas, A. L. Silva, Flat start guess homotopy-based power flow method guided by fictitious network compensation control, Int. J. Electr. Power Energy Syst. 142 (2022) 108311.

[5] A. Bakirtzis, P. Biskas, A decentralized solution to the dc-opf of interconnected power systems, IEEE Transactions on Power Systems 18 (2003) 1007–1013.

[6] S. Cvijic, P. Feldmann, M. Hie, Applications of homotopy for solving AC power flow and AC optimal power flow, in: 2012 IEEE Power and Energy Society General Meeting, IEEE, 2012, pp. 1–8.

[7] V. Ajjarapu, C. Christy, The continuation power flow: a tool for steady state voltage stability analysis, IEEE transactions on Power Systems 7 (1992) 416–423.

[8] F. Milano, Continuous newton's method for power flow analysis, IEEE Transactions on Power Systems 24 (2008) 50–57.

[9] H. Zhao, J. Zhao, J. Qiu, G. Liang, Z. Y. Dong, Cooperative Wind Farm Control With Deep Reinforcement Learning and Knowledge-Assisted Learning, IEEE Transactions on Industrial Informatics 16 (2020) 6912–6921.

[10] N. Sadeghianpourhamami, N. Refa, M. Strobbe, C. Develder, Quantitive analysis of electric vehicle flexibility: A data-driven approach, Int. J. Electr. Power Energy Syst. 95 (2018) 451–462.

[11] Y. Zhou, B. Zhang, C. Xu, T. Lan, R. Diao, D. Shi, et al., A data-driven method for fast ac optimal power flow solutions via deep reinforcement learning, J. Mod. Power Syst. Clean Energy 8 (2020) 1128–1139.

[12] Y. Zhou, W. Lee, R. Diao, D. Shi, Deep reinforcement learning based real-time ac optimal power flow considering uncertainties, J. Mod. Power Syst. Clean Energy 10 (2022) 1098–1109.

[13] K. Baker, Learning warm-start points for ac optimal power flow, arXiv:1905.08860 [eess, math] (2019).

[14] R. Canyasse, G. Dalal, S. Mannor, Supervised learning for optimal power flow as a real-time proxy, in: 2017 IEEE Power & Energy Society Innovative Smart Grid Technologies Conference (ISGT), IEEE, 2017, pp. 1–5.

[15] F. Diehl, Warm-starting AC optimal power flow with graph neural networks, in: 33rd Conference on Neural Information Processing Systems (NeurIPS 2019), 2019, pp. 1–6.

[16] X. Pan, M. Chen, T. Zhao, S. H. Low, Deepopf: A feasibility-optimized deep neural network approach for ac optimal power flow problems, arXiv:2007.01002 [cs, eess] (2021).

[17] G.-F. Fan, L.-Z. Zhang, M. Yu, W.-C. Hong, S.-Q. Dong, Applications of random forest in multivariable response surface for short-term load forecasting, Int. J. Electr. Power Energy Syst. 139 (2022) 108073.

[18] Y.-J. Lin, Explaining critical clearing time with the rules extracted from a multilayer perceptron artificial neural network, Int. J. Electr. Power Energy Syst. 32 (2010) 873–878.

[19] O. L. Santos, D. Dotta, M. Wang, J. H. Chow, I. C. Decker, Performance analysis of a dnn classifier for power system events using an interpretability method, Int. J. Electr. Power Energy Syst. 136 (2022) 107594.

[20] J. Carpentier, Optimal power flows, Int. J. Electr. Power Energy Syst. 1 (1979) 3–15.

[21] R. Nellikkath, S. Chatzivasileiadis, Physics-informed neural networks for ac optimal power flow, Electr. Power Syst. Res. 212 (2022) 108412.

[22] Z. Wang, J.-H. Menke, F. Schäfer, M. Braun, A. Scheidler, Approximating multi-purpose ac optimal power flow with reinforcement trained artificial neural network, Energy AI 7 (2022) 100133.

[23] S. Zhang, D. Zhang, J. Qiao, X. Wang, Z. Zhang, Preventive control for power system transient security based on xgboost and dcopf with consideration of model interpretability, CSEE Journal of Power and Energy Systems 7 (2020) 279–294.

[24] M. Li, W. Wei, Y. Chen, M.-F. Ge, J. P. S. Catalão, Learning the optimal strategy of power system operation with varying renewable generations, IEEE Transactions on Sustainable Energy 12 (2021) 2293–2305.

[25] S. Clémençon, N. Vayatis, Tree-based ranking methods, IEEE Transactions on Information Theory 55 (2009) 4316–4336.

[26] A. Abedi, M. R. Hesamzadeh, F. Romerio, An ACOPF-based bilevel optimization approach for vulnerability assessment of a power system, Int. J. Electr. Power Energy Syst. 125 (2021) 106455.

[27] I. Monedero, F. Biscarri, C. León, J. I. Guerrero, J. Biscarri, R. Millán, Detection of frauds and other non-technical losses in a power utility using Pearson coefficient, Bayesian networks and decision trees, Int. J. Electr. Power Energy Syst. 34 (2012) 90–98.

[28] R. D. Zimmerman, C. E. Murillo-Sánchez, R. J. Thomas, Matpower: Steady-state operations, planning, and analysis tools for power systems research and education, IEEE Transactions on power systems 26 (2010) 12–19.