Multiclass Learning-aided Temporal Decomposition and Distributed Optimization for Power Systems

Farnaz Safdarian, *Member, IEEE,* Amin Kargarian, *Senior Member, IEEE,* Fouad Hasan, *Student Member, IEEE*

***Abstract*— Temporal decomposition is a potential approach to relieve the computation cost of power system multi-interval scheduling problems, such as economic dispatch. In this form of decomposition, the considered scheduling horizon is partitioned into several subhorizons. A subproblem is formulated for each subhorizon, and a distributed optimization algorithm strategy is used to coordinate subproblems. The main existing challenge is decomposing the scheduling horizon to gain the most time saving from distributed computing. This paper serves as an extension to our previous work and presents a machine learning-aided temporal decomposition strategy to partition a scheduling horizon optimally. We have found that the load profile, known before solving economic dispatch, significantly affects the best number of subhorizons. We have used load profiles as inputs to a learner whose goal is to assign a temporal decomposition class to each load profile. Possible decomposition classes are divisors of the considered scheduling horizon. Thus, the proposed learning procedure is a multiclass classification. We have selected Extreme Gradient Boosting that is a tree-based classification learner. Simulation results using real-world load profiles show the promising performance of the proposed algorithm.**

***Index Terms*— Economic dispatch, distributed optimization, temporal decomposition, multiclass classification learner.**

I. INTRODUCTION

LARGE optimization problems are frequently solved for power systems operation and analysis of electricity markets. Many of these problems are multi-interval optimization with intertemporal constraints [1]. The size of optimization problems depends on the system's size and the length of the considered scheduling horizon. Growing the scheduling horizon length increases the computational burden significantly and might make solving the problem in a required time span impossible. Various techniques are applied to reduce the computational complexity of multi-interval scheduling problems and make them solvable in a reasonable time span. Distributed computing is one of these techniques whose objective is to decompose a problem into several smaller subproblems and to apply distributed optimization strategies to coordinate subproblems in a parallel manner [2, 3].

Geographical decomposition and coordination approaches are presented in the literature for two main reasons, i) partition optimization problems according to the power system geographical areas and solve them faster than centralized methods [4-9], and ii) coordinate optimal solution of autonomous entities in the context of multi-agent systems [4]. In [10-12], we have developed temporal decomposition strategies whose objective is to relieve the computational complexity originated from intertemporal constraints. We have discussed that temporal decomposition complements geographical decomposition to further reduce the computational burden of multi-interval optimization problems, such as security-constrained economic dispatch (SCED) and security-constrained unit commitment (SCUC). The temporal decomposition[1] is successfully applied to reduce the solution time of SCED and SCUC [10-12]. Also, it is combined with geographical decomposition in [12].

The way that the power system is partitioned and the number of subproblems significantly impact geographical decomposition's performance and distributed optimization's convergence behavior [13-15]. Increasing the number of subproblems does not necessarily reduce the overall solution time since more subproblems result in more shared variables, and this increases the required number of coordination algorithm iterations to converge. Randomly decomposing a system into several zones may lead to a set of subproblems whose coordination takes many iterations. This may result in not obtaining the best possible outcome of decomposition or even a solution time that is more than that of centralized optimization. In [10-12], we have reported a similar challenge for temporal decomposition. The number of subhorizons and breaking intervals affects the performance of temporal decomposition and the number of distributed algorithm iterations.

In geographical decomposition, the grid can be modeled as a graph with buses and lines representing graph nodes and edges, respectively. Graph-based techniques such as spectral clustering can be applied to partition the grid based on geographical areas so that the least flow is cut between areas [13-15]. Geographical decomposition strategy, however, does not relieve the computational complexity originated from intertemporal constraints. For temporal decomposition, no equivalent graph has been modeled so far. Temporal constraints are interrelated throughout the scheduling horizon, not only neighboring time periods but also non-neighboring periods. Intuitively, temporal partitioning seems more complicated than geographical partitioning. Optimal temporal partitioning

This work was supported by National Science Foundation under Grant ECCS-1944752.

The authors are with the Electrical and Computer Engineering Department, Louisiana State University, Baton Rouge, LA 70803, (e-mail: fsafda1@lsu.edu, kargarian@lsu.edu, fhasan1@lsu.edu).

1. We interchangeably use the terms temporal decomposition, time partitioning, and temporal partitioning.

depends on many factors. Power demand, the rate of change of load between two consecutive time intervals, ramp limits of generating units, minimum on/off time, and characteristics of the considered system are important features affecting the optimal time partitioning for the SCED and SCUC problems. These features affect the number of active intertemporal consistency constraints (e.g., thermal units ramp up/down limits and minimum on/off time) between consecutive subhorizons and the difference between shared variables values from the perspective of neighboring subhorizons. If intertemporal consistency constraints are active, more iterations are required for the coordination algorithm to converge. Therefore, it is desired to select the number of subhorizons so that most of the consistency constraints are not active. In addition, the number of available computing processors and their strength are two other factors affecting the best number of subhorizons. If the available processors are powerful, reducing the size of subhorizons beyond a certain limit does not significantly save time. If processors are not powerful, having more subproblems may be wise. A combination of these features determines the best number of subhorizons in temporal decomposition. However, the status of constraints and shared variables' values are not known before solving the SCED and SCUC.

Recently, machine learning applications to solve various problems have seen increased interest. The behavior of complex phenomena in power systems can be modeled using either simulation or historical datasets. Regression and classification learners read input data and project them to a set of output features. Classification methods have been applied to solving a variety of power system optimization problems. In [16], classification algorithms predict optimal power flow solutions in real-time instead of solving an optimization problem. A classification-based method is presented in [17] to predict the parameters of an environmental multi-objective economic dispatch problem. In [18], classification is used to predict the on/off status of generators in unit commitment. An augmented Lagrangian Hopfield network is used in [19] to enhance the unit commitment solution procedure. A learning-based method is proposed in [20] to determine the duration of aggregated chronological time periods of a centralized unit commitment problem using non-supervised hierarchical clustering techniques. Classification is used to de-commit extra spinning reserve units caused by minimum uptime/downtime constraints. Although classification learners have been widely used for power system optimization, their applications for optimal decomposition (neither temporal nor geographical) and distributed optimization have not been explored despite their potential advantages to enhance decomposition. This has motivated us to perform this study.

In this paper, a learning-aided temporal decomposition approach is proposed to determine the best time partitioning scheme that results in the best performance of distributed optimization for solving the security-constrained economic dispatch problem. To the best of our knowledge, this paper is the first study of using learning techniques for optimal time decomposition. The considered scheduling horizon is partitioned into several subhorizons based on the method presented in [11]. We study the effect of the number of subhorizons on temporal decomposition performance and analyze the solution time, the number of iterations, and solution accuracy. The possible decomposition schemes belong to the scheduling horizon's divisors, yielding the same-sized subhorizons to take the best advantage of parallel computing. We propose modeling the optimal temporal partitioning approach as a multiclass classification whose input is the load profile known before solving SCED. The classifier's output is the best number of subhorizons. We have used Extreme Gradient Boosting (XGBoost) as the learner, a tree-based classifier suitable for multiclass classification. Once the learner is trained, validated, and tested offline, it will be used to identify the best time decomposition class for a given load profile before solving SCED. We have also tested the learning-aided approach on unit commitment. Simulation results show the promising performance of the proposed decomposition approach.

The difference between this paper and [10-12] lies in presenting a systematic algorithm for optimal time partitioning. Although temporal decomposition and coordination strategies are presented in [10-12], trial and error were used to partition the scheduling horizon. Partitioning is one of the main barriers to temporal decomposition applications in power systems. This paper solves this challenge using a learning-aided strategy. First, the strategies presented in [10-12] are applied for offline data preparation. A learner is then trained to map a demand profile to its corresponding best decomposition class. To solve a new SCED problem, the learner determines the optimal class first, and then the coordination algorithm in [10] is used to obtain the most time saving by distributed optimization.

The remainder of this paper is organized as follows. Temporal decomposition is briefly described in Section II. Several motivating examples are presented in Section III, and important factors for temporal decomposition are analyzed. The proposed learning-aided time partitioning algorithm is presented in Section IV. Numerical results are discussed in Section V, and concluding remarks are provided in Section VI.

II. Time Decomposition Strategy

Consider solving the SCED problem in the Appendix for a scheduling horizon of T time intervals, as shown in Fig. 1.a. The horizon can be decomposed into N subhorizons, each consisting of a subset of time periods, as depicted in Fig. 1.b. The length of all time periods in Fig. 1 is the same. An SCED subproblem can be formulated for each subhorizon. The subproblems are connected through ramping up and down of generating units, which are intertemporal constraints. The computational burden of solving SCED for each subhorizon is less than that of the whole scheduling horizon.

SCED subproblems can be formulated regardless of their correlation with one another. Subproblems can then be solved in parallel. However, this approach does not provide a feasible solution as the intertemporal constraints for transition between subhorizons are ignored. Another approach is to start from the first subhorizon and solve subproblems sequentially by fixing values obtained in the last interval of a subproblem n as the

initial state for subproblem $n+1$. This approach provides a feasible but suboptimal solution.

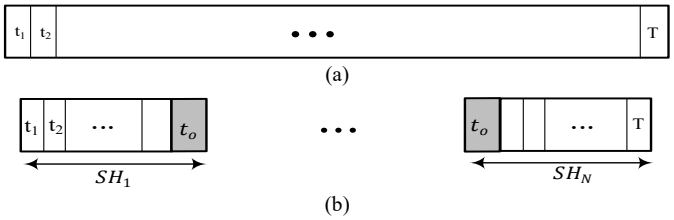

Fig. 1. a) Overall scheduling horizon with T time intervals and b) decomposition of overall horizon into N subhorizons (SHs).

As described in [11], the concept of overlapping (or coupling) intervals is introduced to allow solving subproblems in parallel while ensuring optimality of the obtained solution. The first interval of each subhorizon n is duplicated and added in subhorizon $n-1$ as its last interval, as shown with t_o (called overlapping time interval) in Fig. 1.b. Variables and constraints of the overlapping time interval appear in both subproblems corresponding to subhorizons $n-1$ and n.

Power produced by generating units in overlapping time intervals is shared between consecutive subhorizons. Consider two consecutive subhorizons (subproblems) $n-1$ and n. Power produced by unit u at the overlapping time interval t_o is named, respectively, p_{ut_o} and p'_{ut_o} from the perspective of subhorizons $n-1$ and n. Since these shared variables are physically the same, the following consistency constraint must be satisfied to ensure the feasibility of results from the whole scheduling horizon's perspective.

$$CC: p_{ut_o} - p'_{ut_o} = 0 \qquad \forall u \tag{1}$$

This consistency constraint can be relaxed using the concept of augmented Lagrangian relaxation [21]. Using the concept of Nesterov momentum for gradient descent acceleration [22, 23], an accelerated auxiliary problem principle (A-APP) is presented in [11] to coordinate subproblems iteratively and obtain the SCED solution. This approach, which allows the parallel solution of SCED subproblems, is presented below for two subproblems $n-1$ and n. The objective function (2) is to minimize the generation cost of units in time periods corresponding to subproblem $n-1$ plus the three penalty terms related to consistency constraint relaxation. The objective function of subproblem n is formulated by (3).

$SCED_{n-1}$ at iteration k given $\hat{p}'^{\,k-1}_{ut_o}$:

$$\min \sum_t \sum_u a p_{ut}^2 + b p_{ut} + c_{ut} \tag{2}$$
$$+ \sum_u \frac{\rho}{2} \left\| p_{ut_o} - \hat{p}^{k-1}_{ut_o} \right\|^2 + \hat{\tau}^{\dagger} p_{ut_o} + \mu p^{\dagger}_{ut_o} \left(\hat{p}^{k-1}_{ut_o} - \hat{p}'^{\,k-1}_{ut_o} \right)$$

Subject to equipment (e.g., generating units) and network (e.g., power flow) constraints corresponding to subhorizon $n-1$

$SCED_n$ at iteration k given $\hat{p}^{k-1}_{ut_o}$:

$$\min \sum_t \sum_u a p_{ut}^2 + b p_{ut} + c_{ut} \tag{3}$$
$$+ \sum_u \frac{\rho}{2} \left\| p'_{ut_o} - \hat{p}'^{\,k-1}_{ut_o} \right\|^2 - \hat{\tau}^{\dagger} p'_{ut_o} + \mu p'^{\,\dagger}_{ut_o} \left(\hat{p}'^{k-1}_{ut_o} - \hat{p}^{k-1}_{ut_o} \right)$$

Subject to equipment and network constraints corresponding to subhorizon n.

A-APP tuning parameters are ρ and μ. Multiplier $\hat{\tau}$ and variables $\hat{p}_{ut_o}$ and $\hat{p}'_{ut_o}$, which are the modified forms of Lagrange multiplier τ and shared variables p_{ut_o} and p'_{ut_o} using Nesterov momentum, are iteratively updated as follows [10, 11].

$$\tau^{k+1} = \tau^k + \beta \left(p^k_{ut_o} - p'^{\,k}_{ut_o} \right) \tag{4}$$

$$\alpha_{k+1} = \frac{1 + \sqrt{1 + 4\alpha_k^2}}{2} \tag{5}$$

$$\hat{p}^{k+1}_{ut_o} = p^k_{ut_o} + \frac{\alpha_k - 1}{\alpha_{k+1}} \left(p^k_{ut_o} - p^{k-1}_{ut_o} \right) \tag{6}$$

$$\hat{p}'^{\,k+1}_{ut_o} = p'^{\,k}_{ut_o} + \frac{\alpha_k - 1}{\alpha_{k+1}} \left(p'^{\,k}_{ut_o} - p'^{\,k-1}_{ut_o} \right) \tag{7}$$

$$\hat{\tau}^{k+1} = \tau^k + \frac{\alpha_k - 1}{\alpha_{k+1}} (\tau^k - \tau^{k-1}) \tag{8}$$

where β is a suitable positive step-size, and α is the momentum coefficient. Tuning parameters and step sizes can be individually determined for each shared variable. The above procedure is repeated until $\left| p_{ut_o} - p'_{ut_o} \right| \le \epsilon$. This algorithm can be extended for different numbers of subhorizons. We refer to [11, 24] for more details on the problem formulation and solution algorithm.

III. Important Factors and Motivating Examples

Reducing the length of subhorizons and the number of coordination algorithm iterations reduces the distributed optimization solution time. The less the number of variables and constraints of a subproblem is, the less the computational time would be. However, because of increasing the number of shared variables, the required number of iterations of the distributed algorithm to coordinate subproblems and the total solution time might increase. Moreover, partitioning the problem from time intervals with a considerable number of active intertemporal ramping up and down constraints results in a larger gap between shared variables in the first few iterations. This might increase the required number of iterations of distributed optimization to converge. The scheduling horizon must be decomposed carefully to obtain the best solution time.

In this section, we illustrate a few examples to show the necessity of optimal temporal decomposition and to investigate the factors that affect it. The considered scheduling problem is a week-ahead SCED. The overall horizon is fixed, but the length of subhorizons is studied. The objective function is to minimize the generation cost subject to thermal unit constraints, power balance equalities, line flow limits, and voltage angle limitations under normal and contingency conditions.

A. Motivating Examples

A 3-Bus System with Smooth Load Profile: The maximum load variation between each two consecutive time interval is less than the ramping limitations of generating units. Therefore, intertemporal consistency constraints connecting subhorizons, i.e., ramp up and down constraints, are not active. This makes subproblems loosely connected and yields small differences in the desired values of shared variables, i.e., power generated by units at overlapping time intervals, from the perspective of neighboring subproblems. As shown in Fig. 2, by increasing the number of subproblems, the size of each subproblem will be smaller, and the overall solution time decreases. However, since the system is small, decomposing the problem beyond 20 subhorizons does not significantly save time. It also needs more computational resources and may slightly increase the solution error as compared to the centralized method. Hence, we suggest not decomposing the scheduling horizon beyond 20 subhorizons.

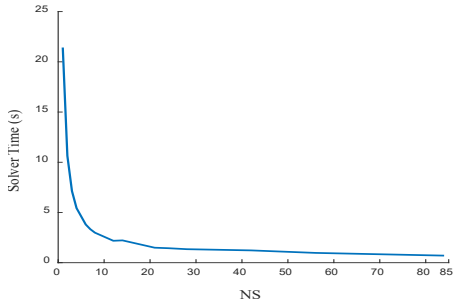

Fig. 2. Overall solution time versus number of subhorizons (NS) for the 3-bus test system with a flat load profile.

B. IEEE 24-Bus System with Variable Load Profile:

Figure 3a shows the solution time versus the number of subhorizons for a given load pattern, called pattern one [28]. Increasing the number of subhorizons reduces the solution time; however, increasing the number of subhorizons beyond nine increases the solution time. This is because of having many active intertemporal consistency constraints and large differences in the desired values for power generated by units at overlapping time intervals from the perspective of neighboring subhorizons. Hence, the number of iterations and the overall solution time of the distributed algorithm increase. We have reduced the load by 5% and redrawn the curve. As shown in Fig. 3b, this monotonous load decrease does not change the curve pattern.

We have tested another load pattern, called pattern two [28]. The solution time does not follow a curve similar to Fig. 3.a and has a non-monotonic behavior. This is because of the sophisticated behavior of units' ramp up/down constraints. These intertemporal constraints connect intervals $\{1, \dots, T\}$ and will be active depending on the load pattern and system characteristics. This results in an unpredictable pattern in the desired shared variable values from the perspective of neighboring subhorizons and a non-monotonic behavior in the solution time pattern. The load is reduced by 5%, and the curve is plotted in Fig. 4b, which is similar to Fig. 4a. Comparing Figs. 3 and 4 show that the load profile pattern has a more significant impact on the solution time than a small load increase or decrease.

C. Important Factors for Solution Time

The factors affecting the overall solution time versus the number of subhorizons include 1) system characteristics, 2) generators characteristics, 3) the number of active ramp up/down constraints for transition between subhorizons, and 4) the desired values of power generated by units at overlapping time intervals from the perspective of neighboring subhorizons. For a given system with a set of generating units, the third and fourth factors should be analyzed to obtain the optimal temporal decomposition. However, they are unknown before solving the problem. The load profile plays a critical role in the status of intertemporal constraints and the values of variables. Hence, the load profile can be used to analyze the number of subhorizons versus the solution time and find the best temporal decomposition scheme.

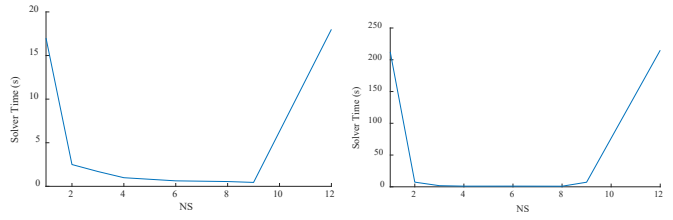

Fig. 3. Overall solution time versus number of subhorizons (NS) for the IEEE 24-bus system with a) load pattern one and b) load pattern one with a 5% decrease.

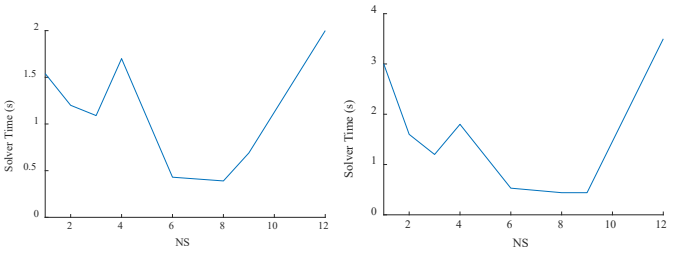

Fig. 4. Overall solution time versus number of subhorizons (NS) for the IEEE 24-bus system with a) load pattern two and b) load pattern two with a 5% decrease.

IV. Proposed Learning-Aided Methodology

Given that the load profile is known before solving SCED, we propose a learning-aided algorithm for time partitioning. The goal of this learner is to project the best number of subhorizons to the load profile pattern. As illustrated in Fig. 5, the input to the multiclass classifier is the load profile over the considered scheduling horizon, and its output is the best time partitioning scheme.

A. Offline Data Labelling

Historical and predicted system load profile patterns for the considered scheduling horizon can be collected. For each load pattern lp, all divisors of the considered scheduling horizon are determined as the possible decomposition classes (denoted by Ω_{cl}) with subhorizons with equal length. For a scheduling

horizon with 72 intervals, for instance, Ω_{cl}={1, 2, 3, 4, 6, 7, 8, 9, 12, 18, 24, 36, 72}. Subhorizons with different lengths can also be considered. However, we suggest subhorizons with the same length that yields almost the same sized optimization subproblems with similar solution times. This results in gaining the most advantage of parallel computing with minimum CPU idle time.

A-APP is applied to solve SCED in a distributed manner for each decomposition class cl. An error-time index is created by combining the solution time and the relative error to determine the best class for each load pattern.

$$\varphi_{cl} = \omega_1 \times rel_{cl} + \omega_2 \times CPU_{time,cl} \quad \forall cl \in \Omega_{cl} \quad (9)$$

The rel index is the relative error between the optimal costs obtained by centralized (f^*) and distributed (f_{cl}^d) approaches. Parameter $CPU_{time,cl}$ is the solution time of each class cl.

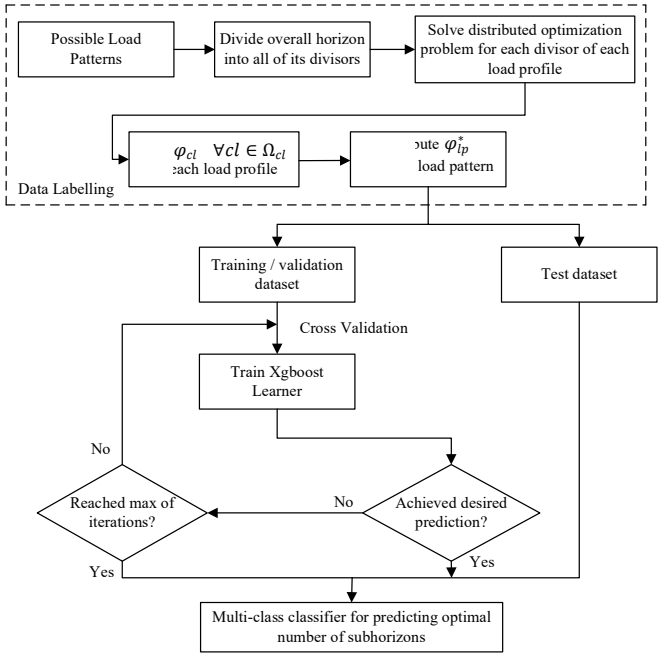

Fig. 5. Flowchart of the proposed learner-based temporal decomposition algorithm.

$$rel_{cl} = \left| \frac{f^* - f_{cl}^d}{f^*} \right| \quad (10)$$

Weighting factors ω_1 and ω_2 assign priority to rel_{cl} and $CPU_{time,cl}$ indices. A system operator can determine the weight values based on its preference for solution time and accuracy. After solving distributed SCED for each cl of load pattern lp, the decomposition class with the smallest error-time index is determined (denoted by cl_{lp}^*), and the load pattern is assigned to this class.

$$\varphi_{lp}^* = \min\{\varphi_{cl} \quad \forall cl \in \Omega_{cl}\} \quad (11)$$

This offline procedure, whose pseudocode is shown in Data Labelling Algorithm, assigns the best number of subhorizons as the class label for each load profile.

Data Labelling Algorithm Pseudocode for multiclass classification of load patterns and data preparation

1: Read historical load patterns for the considered scheduling horizon
2: Determine weighting factors ω_1 and ω_2
3: **Do** for all possible load patterns
4: **for** $cl \in \Omega_{cl}$
5: Decompose the considered horizon into cl equal subhorizons
6: **while** $|CC| > \epsilon$, $k = k + 1$ **do**
7: Solve SCED subproblems in parallel and determine optimal values of $p_{ut_o}^k$ and $p_{ut_o}^{\prime k}$
8: Exchange $p_{ut_o}^k$ and $p_{ut_o}^{\prime k}$ between SCED subproblems
9: Update τ^k by (4)
10: Calculate α_{k+1} by (5)
11: Update $\hat{p}^{k+1}$, $\hat{p}'^{k+1}$, and $\hat{\tau}^{k+1}$ by (6)-(8)
12: **end while**
13: Record $CPU_{time,cl}$ and calculate rel_{cl}
14: Calculate $\varphi_{cl} = \omega_1 \times rel_{cl} + \omega_2 \times CPU_{time,cl}$
15: **end for**
16: Determine φ_{lp}^* for each load pattern
17: Assign the load pattern to cl_{lp}^*

Remark 1: If energy storage and unit commitment constraints are considered in the optimization problem, the modeling and coordination strategies in [10, 12] need to be applied.
Remark 2: The learning dataset should be updated every several months or years after installing new generating units to adapt the classifier to changes in the system characteristics.

B. Multiclass Learning for Temporal Decomposition

We need a learner to project each load profile lp to its best decomposition class cl_{lp}^*. The input to this learner is a time-varying demand vector, and the output is an integer, which is limited among the divisors of the considered scheduling horizon. A multiclass classification learner can structure this learning procedure.

Selecting Learner: Various algorithms exist to form a multiclass classifier, among which tree-based algorithms are widely used as their results can be interpreted properly. Decision tree, random forest, and Extreme Gradient Boosting (XGBoost) are among the most popular and efficient tree-based approaches [25]. Classification trees are made of nodes, which separate data based on some impurity criteria, and leaves that determine classes. In the decision tree, each node is selected based on a characteristic that provides the best split with the least impurity and the most information gain. To select a certain characteristic to split a node, the information gain by splitting on that node is calculated and the split with the largest gain is made. In the random forest, a random number of characteristics is selected at each step, and different decision trees are made based on those characteristics. A class receiving the most votes from decision trees is determined as the final class. Trees of a random forest are independent of each other.

The chronological order of time intervals must be maintained in time partitioning. Thus, the decision tree and the random forest are not suitable for time partitioning as they randomly select trees and do not ensure maintaining the chronological order of time intervals. XGBoost maintains the chronological order of time intervals. In XGBoost, trees are made based on regression, and the predicted value is updated

after each regression tree is made until a suitable prediction is made. XGBoost is a combination of gradient boosting, regularization, unique regression trees, approximate greedy algorithm, weighted quantile sketch, sparsity-aware split finding, parallel learning, cache-award access, and blocks for out-of-core computation [26, 27]. As compared to other gradient boosting methods, XGBoost has two additional features to prevent over-fitting. The weights of a new tree can be scaled down by a given constant to reduce the impact of a single tree on the final score and to provide an opportunity for the next trees to improve the model. XGBoost also performs better than other tree boosting methods. This is mainly because it supports an approximate split finding, which improves building trees and scales well with the number of CPU cores. Thus, we have used XGBoost for optimal temporal partitioning projection.

XGBoost Model for Load Profile Classification: Assuming M and K denote, respectively, the number of rounds of XGBoost and the number of trees, a total number of $M \times K$ decision trees are generated. XGBoost pre-sorts attributes and greedily finds the split point with the largest information gain [26]. Assume a dataset $\mathcal{D} = \{(P_{lp}, \varphi^*_{lp}): lp = 1, \dots, N_s\}$ where P_{lp} is the demand vector lp over the scheduling horizon whose best decomposition class based on (11) is φ^*_{lp}. We define $\widehat{\varphi^*}$ as predicted classes by the learner.

$$\widehat{\varphi^*_{lp}} = \sum_{k=1}^{K} f_k(P_{lp}) \tag{12}$$

$f_k(\cdot)$ is a regression tree, and $f_k(P_{lp})$ represents the score given by the kth tree to lpth observation. If the regression tree f_k for $k = 1, \dots, K$ is achieved, expression (12) will provide the predicted temporal decomposition class. Thus, the goal of training the learner is to find the optimal regression trees (denoted by $f^*_k(P_{lp})\ \forall k$) that minimize the following regularized objective function.

$$\mathcal{O} = \sum_{lp} l(\varphi^*_{lp}, \widehat{\varphi^*_{lp}}) + \sum_{k} \gamma T + \frac{1}{2}\lambda \|f_k\|^2 \tag{13}$$

where $l(\cdot)$ is the loss function quantifying prediction quality. XGBoost uses this loss function to build trees by minimizing $\mathcal{O}$. Parameters γ and λ control penalty for the number of terminal nodes or leaves (T). Parameter γ encourages pruning trees. The second term of (13) is added to prevent over-fitting. This term simplifies models produced by the learner.

An iterative method is used to minimize the objective function. At iteration $j = 0$, the initial guess for each class's probability is one divided by the number of classes. At iteration $j \neq 0$, XGBoost builds a tree by finding the output value for a leaf f_j that minimizes the following objective function.

$$\mathcal{O}^j = \sum_{lp=1}^{N_s} l(\varphi^*_{lp}, \hat{\varphi}^{*(j-1)}_{lp} + f_j(P_{lp})) + \sum_{j} \gamma T + \frac{1}{2}\lambda \|f_j\|^2 \tag{14}$$

Using the second-order Taylor expansion, this function is simplified and a formula for loss reduction is derived by solving for the optimal value.

$$\mathcal{O}^j = \sum_{lp=1}^{N_s} l(\varphi^*_{lp}, \hat{\varphi}^{*(j-1)}_{lp}) + g_{lp} f_j(P_{lp}) + \frac{1}{2} h_{lp\,j} f_j^2(P_{lp}) + \sum_{j} \gamma T + \frac{1}{2}\lambda \|f_j\|^2 \tag{15}$$

where functions g_{lp} and h_{lp} are defined as follows:

$$g_{lp} = \partial_{\hat{\varphi}^{*(j-1)}_{lp}} l\left(\varphi^*_{lp}, \hat{\varphi}^{*(j-1)}_{lp}\right) \tag{16}$$

$$h_{lp} = \partial^2_{\hat{\varphi}^{*(j-1)}_{lp}} l\left(\varphi^*_{lp}, \hat{\varphi}^{*(j-1)}_{lp}\right) \tag{17}$$

Function $\mathcal{O}$ is solved based on the output value $f^*_k(P_{lp}), \forall k$ for the leaf to create XGBoost trees. Since the goal is to find the output value that minimizes the objective function, derivatives are calculated based on the output value and are made equal to zero. The desired output value is achieved as:

$$f^*_k(P_{lp}) = -\frac{1}{2}\sum_{k=1}^{K} \frac{\left(\sum_{lp\in\Omega_{lp}} g_{lp}\right)^2}{\sum_{lp\in\Omega_{lp}} h_{lp} + \lambda} + \gamma T \tag{18}$$

After the leaf (node) output value is calculated, the best split at the given node is found based on the gain and similarity score to grow the XGBoost tree. Then gain ($\mathcal{g}$) is calculated as the sum of the left and right leaves' similarity scores minus the root's similarity score. The best split at any given node is the split with the largest information gain.

$$\mathcal{g} = \frac{1}{2}\left[\frac{(\sum_{lp\in\Omega^L_{lp}} g_{lp})^2}{\sum_{lp\in\Omega^L_{lp}} h_{lp} + \lambda} + \frac{(\sum_{lp\in\Omega^R_{lp}} g_{lp})^2}{\sum_{lp\in\Omega^R_{lp}} h_{lp} + \lambda} - \frac{(\sum_{lp\in\Omega_{lp}} g_{lp})^2}{\sum_{lp\in\Omega_{lp}} h_{lp} + \lambda}\right] - \gamma \tag{19}$$

where Ω_{lp} is the set of load profiles in the current node, and the sets of observations available in the left and right leaves after the split are denoted by Ω^L_{lp} and Ω^R_{lp}, respectively.

The trained XGBoost can be used, as shown in Fig. 6. The learner reads the load profile and decomposes the scheduling horizons into the best number of subhorizons (SH).

Fig. 6. Utilizing procedure of the trained XGBoost learner.

V. Numerical Results

Four cases are studied, A) SCED for the IEEE 118-bus system without wind power, B) SCED for the IEEE 118-bus system with 30% wind power penetration, C) SCED for a 2006-bus system, and D) unit commitment for the IEEE 24-bus system. System information and characteristics are given in [28]. Simulations are carried out on a personal computer with a 3.70 GHz Intel(R) Xeon(R) CPU and 16 GB of RAM. The Yalmip toolbox in Matlab and Gurobi is used to solve optimization problems, and Python learning toolboxes XGBoost and sklearn are used to train multiclass classification learners [27, 29].

A. IEEE 118-Bus System without Wind Power

Dataset Preparation: The considered scheduling problem is a week-ahead economic dispatch. To show the algorithm's performance for practical load patterns, we have extracted 6669 real-world weekly load profiles from PJM [30]. These load profiles are scaled to be used for the IEEE 118-bus system. Figure 7 shows some random samples of load profiles. A load profile can belong to any divisors of 168 as the possible classes. For each load profile and each divisor, the distributed algorithm is implemented. The best divisor that provides φ_{lp}^{*} is assigned to each load profile as its decomposition class. This offline procedure provides train and test datasets.

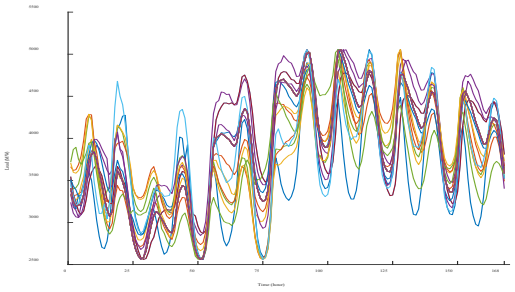

Fig. 7. Some random samples of PJM load profiles [30], shown by different colors, that are normalized for the IEEE 118-bus system.

We have observed that all load profiles in the train and test datasets belong to $\{2, 4, 7, 8, 12\}$ classes of divisors of 168. No load profile is assigned to other divisors. We keep these five classes and drop others. We define the decomposition classes as $cl_1 = 2$, $cl_2 = 4$, $cl_3 = 7$, $cl_4 = 8$, and $cl_5 = 12$. For example, $cl_2 = 4$ means that decomposition class two includes load profiles with the best number of subhorizons equals 4. Although no significant imbalance is observed in the dataset, various approaches can mitigate possible dataset imbalance effects [31].

Temporal Decomposition Classifier Training: Eighty percent of the dataset is selected randomly for training and twenty percent for testing. The softmax function is used to normalize the probability distribution of predicted output classes so that the sum of all probabilities becomes one. The maximum tree depth D, the number of trees K, and regularization parameters such as learning rate L, γ, and λ need to be tuned to train XGBoost. These parameters have been determined based on sensitivity analysis and preliminary investigation of their acceptable ranges [26, 32]. Parameters γ and λ are set to one while tuning other hyper-parameters. We set the parameter D, which controls the sequential process of growing trees, in a range of $D \in \{1, 2, \dots, 6\}$. It is suggested not to exceed the depth of a tree more than 6. Parameter L should be in the range of $0 < L < 1$. The accuracy of the learner increases by increasing K; however, this may cause overfitting. We have tested various values for this parameter as $K \in \{1, 2, \dots, 10\}$. Table I shows the learner's accuracy for different parameters. We have found the best combination for hyper-parameters as $L = 0.5$, $D = 6$, and $K = 6$. Except for $D = 1$ (which is not suggested for a multiclass classification), if parameters are selected from their acceptable ranges, high accuracy is achieved, and the results are not much sensitive to the choice of hyperparameters.

TABLE I
ACCURACY OF PARAMETER TUNING

Accuracy	D	K	L	Rounds
0.57	1	4	0.1	1
0.85	2	4	0.1	1
0.93	3	4	0.1	1
0.98	4	4	0.1	1
0.98	5	4	0.1	1
0.98	6	4	0.1	1
0.98	4	1	0.1	1
0.98	4	10	0.1	1
0.98	4	4	0.01	1
0.98	4	4	1	1
0.99	4	4	0.1	2
0.99	4	4	0.1	3
0.99	4	4	0.1	4
0.99	6	6	0.5	5

Evaluating Temporal Decomposition Classifier: Twenty percent of the dataset is used for testing the multiclass classification learner. The following four primary indices are introduced for each decomposition class to interpret predicted results and analyze the learning-aided temporal decomposition accuracy.

- *True positives* (TP): a load profile lp is predicted to belong to a decomposition class cl and its actual class is cl.
- *True negatives* (TN): a load profile lp is predicted to not belong to a decomposition class cl and its actual class is not cl.
- *False positives* (FP): a load profile lp is predicted to belong to a decomposition class cl, but its actual class is not cl.
- *False negatives* (FN): a load profile lp is predicted to not belong to a decomposition class cl, but its actual class is cl.

We use classification accuracy, precision, recall, and F_1 score metrics to analyze the quality of the classification leaner.

$$\text{Accuracy}_{cl} = \frac{TP_{cl} + TN_{cl}}{TP_{cl} + TN_{cl} + FP_{cl} + FN_{cl}} \tag{20}$$

Precision is defined as the fraction of true positives out of total instances predicted as positives.

$$Precision_{cl} = \frac{TP_{cl}}{TP_{cl} + FP_{cl}} \tag{21}$$

	Actual cl_1 2 subhorizons	Actual cl_2 4 subhorizons	Actual cl_3 7 subhorizons	Actual cl_4 8 subhorizons	Actual cl_5 12 subhorizons	Total predict	$Precision_{cl}$
Predicted cl_1	664	0	0	2	3	669	99.253%
Predicted cl_2	3	199	0	0	0	202	98.515%
Predicted cl_3	0	0	145	0	0	145	100%
Predicted cl_4	1	2	0	194	0	197	98.477%
Predicted cl_5	0	0	0	0	121	121	100%
Total actual	668	201	145	196	124	1334	
$Recall_{cl}$	99.40%	99.01%	100%	98.98%	97.5%		

Fig. 8. Confusion matrix.

- Recall is defined as the fraction of instances belonging to positive classes that are predicted as positives.

$$Recall_{cl} = \frac{TP_{cl}}{TP_{cl} + FN_{cl}} \tag{22}$$

The F_1 score is the harmonic mean of precision and recall, defined as:

$$F_{1,cl} = 2 \times \frac{precision_{cl} \times recall_{cl}}{precision_{cl} + recall_{cl}} \tag{23}$$

The confusion matrix is represented in Fig. 8. Green blocks show the number of load profiles whose decomposition classes are predicted correctly. Orange blocks depict the number of load profiles that are misclassified. As an example, 668 load profiles in the test dataset belong to cl_1. 664 (99.4%) of those are predicted correctly, and only four (0.6%) load profiles are misclassified. The misclassification percentage of all decomposition classes is low, and almost all the load profiles are projected to their correct best decomposition class. The overall TP and TN are the summations of all correctly classified load patterns, regardless of their classes. The overall FP and FN refer to the overall incorrectly predicted load patterns.

We have also calculated the overall classification accuracy, precision, recall, and F_1 score metrics for all classes combined, as depicted in Table II. The overall accuracy, which is the proportion of correct predictions over all predictions, is more than 99%. The confusion matric and overall indices prove the promising performance of the proposed learning-aided temporal decomposition algorithm.

TABLE II
PERFORMANCE INDICES OF LEARNER FOR 118-BUS SYSTEM WITHOUT WIND POWER

Index	Value
Accuracy	0.99
F_1	0.99
Precision	0.99
Recall	0.99

We have also used a support vector machine and neural networks for classification. The comparison of performance metrics shows that the three classification approaches provide promising results, with a prediction accuracy above 99%. This indicates that several learners may work well for the considered temporal decomposition problems.

Distributed Optimization Results: We have randomly selected a load profile, shown in Fig. 9, from the IEEE 118-bus system's test dataset and have carried out the distributed optimization with different numbers of subhorizons. The operation cost obtained by centralized economic dispatch is \$9,050,971. The relative error for all decomposition classes is less than $2e-6$. We choose $\omega_1 = 1e6$ and $\omega_2 = 1$. Table III illustrates the relative error, solution time, and φ_{cl} for each decomposition class. The solver time versus the number of subhorizons is plotted in Fig. 10. The error-time index for this given load profile is $\varphi^*_{lp} = 0.7327$. Thus, the best strategy for this load pattern is to decompose the considered scheduling horizon into eight subproblems. We have also carried out the classification leaner for this load profile. The predicted decomposition class by the learner is also cl_4, which refers to eight subproblems.

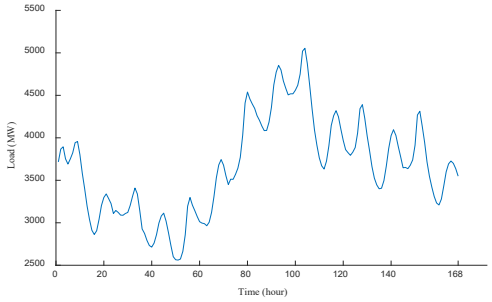

Fig. 9. A load pattern used to test the algorithm [42].

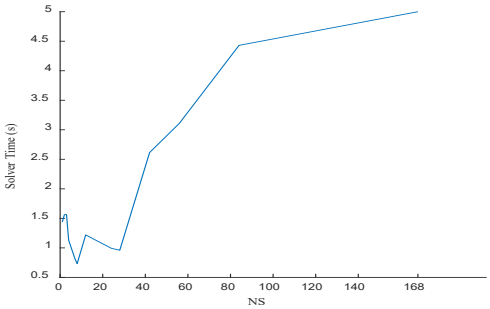

Fig. 10. Solver time versus number of subhorizons for the sample load profile in Fig. 9.

We have implemented a similar procedure for all load profiles in the dataset. The minimum, maximum, and average values of the relative error and the average solver time are reported in Table IV. The best average solution time is obtained for decomposition into 28 subhorizons. However, the best partitioning strategy for each load profile should be determined

independently by the learner before solving distributed optimization.

TABLE III
RELATIVE ERROR, SOLVER TIME, AND ERROR-TIME INDICES FOR DECOMPOSITION CLASSES CORRESPONDING TO LOAD PROFILE IN FIG. 9

Number of subhorizons	rel	Solver time (sec)	φ_{cl}
1	-	1.4	1.4363
2	~0*	1.6	1.5626
3	2e-09	1.6	1.5649
4	~0	1.1	1.1307
6	2e-09	0.9	0.9175
7	2e-08	0.8	0.8138
8	~0	0.7	0.7327
12	3e-08	1.2	1.2497
24	6e-07	1	1.594
28	2e-06	1	2.9599
42	2e-07	2.6	2.8136
56	1e-07	3.1	3.2102
84	5e-07	4.4	4.5298

* Values less than 1e-10 are assumed to be ~0

TABLE IV
MINIMUM, MAXIMUM AND AVERAGE RELATIVE ERROR AND AVERAGE SOLVER TIME OF ALL LOAD PROFILES IN DATA SET FOR 118-BUS SYSTEM

Number of subhorizons	max rel	min rel	Average rel	Average solver time (sec)
1	-	-	-	1.6
2	6e-5	~0	3e-6	1.9
3	6e-6	~0	1e-6	2.6
4	8e-6	~0	8e-7	2
6	9e-6	1e-9	1e-6	1.6
7	4e-5	1e-8	3e-6	1.5
8	4e-6	~0	1e-6	1.4
12	5e-2	3e-8	1e-2	1.7
24	6e-5	6e-7	1e-5	2.5
28	4e-4	2e-6	3e-5	1.2
42	6e-5	2e-7	1e-5	2.6
56	8e-4	1e-7	6e-5	3.1
84	5e-4	5e-7	1e-4	3.9

B. *IEEE 118-Bus System with Wind Power*

It is assumed that 30% of the load is supplied with wind turbines. To consider an extreme impact of wind uncertainty, wind power scenarios for each time interval are generated randomly between 0 to 100% of wind power capacity. Figure 11 shows the net load for the sampled load profiles in Fig. 7 with 30% wind power penetration. The net load, used as input to the classifier, fluctuates significantly due to wind power variations.

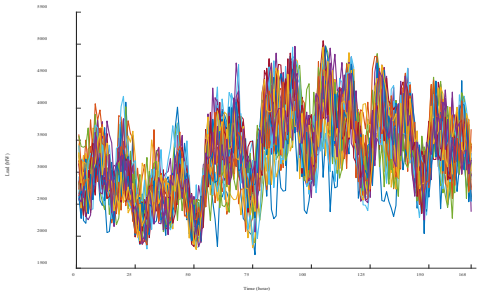

Fig. 11. Net load profiles if Fig. 7 with 30% wind power penetration.

The overall classification accuracy, precision, recall, and F_1 score for all classes combined are reported in Table V. These four indices are around 99%, similar to those for the system without wind power uncertainty.

TABLE V
PERFORMANCE INDICES OF LEARNER FOR 118-BUS SYSTEM WITH WIND POWER

Index	Value
Accuracy	0.99
F_1	0.99
Precision	0.99
Recall	0.99

C. *2006-Bus System*

The classifier accuracy, F_1 score, precision, and recall for this large system are reported in Table VI. These indices are above 98%, showing the promising performance of the trained XGBoost classifier.

TABLE VI
PERFORMANCE INDICES OF LEARNER FOR 2006-BUS SYSTEM

Index	Value
Accuracy	0.99
F_1	0.99
Precision	0.98
Recall	0.98

We have implemented distributed optimization with different decomposition classes for all load profiles. Table VII shows the minimum, maximum, and average values of the relative error and the average solver time. The solver time for the centralized SCED is 142 seconds, which is larger than that of distributed optimization with any decomposition class.

TABLE VII
MINIMUM, MAXIMUM, AND AVERAGE RELATIVE ERROR AND AVERAGE SOLVER TIME OF ALL SAMPLE LOAD PROFILES FOR 2006-BUS SYSTEM

Number of subhorizons	max rel	min rel	Average rel	Average solver time (sec)
1 (centralized)	-	-	-	142
2	~0	~0	~0	67
3	~0	~0	~0	35
4	~0	~0	~0	23
6	1e-7	~0	1e-7	16
7	7e-7	~0	1e-7	13
8	7e-7	~0	1e-7	12
12	6e-7	~0	1e-7	13
24	3e-6	~0	2e-6	36
28	1e-6	~0	1e-6	38

If one uses the average values to find the best decomposition class, any load profile should be decomposed into eight subhorizons. Figures 12 and 13 show solution time and logarithmic relative error histograms if the problem is always decomposed into eight subproblems. And Figs. 14 and 15 are solution time and logarithmic relative error histograms if one uses a learner to find the best decomposition class for each load profile. As observed, the learning-aided approach outperforms in terms of the solution time and relative error. While 44% of the average-based strategy's cases lie in (11.5, 12.5] seconds solution time interval, 48% of the learning-aided approach's samples take (8, 9] seconds. Also, 28% of the average-based

strategy's cases have a relative error in the range of 10^{-7}, the learning-aided approach provides a relative error smaller than 10^{-10} for all samples. The average solution time and relative error of the learning-aided approach are 10.1 and 7×10^{-11}, which roughly 20% and 100% smaller than those of average-based strategy. Only for 4% of cases (highlighted in red in Fig. 14), the decomposition class of average-based and leaning-aided strategies are the same. Although the proposed approach takes approximately 2 seconds more for 24% of cases (highlighted in gray in Figs. 14 and 15), the relative errors of these cases are $\sim 10^{-4}$ smaller than those obtained by the average-based strategy.

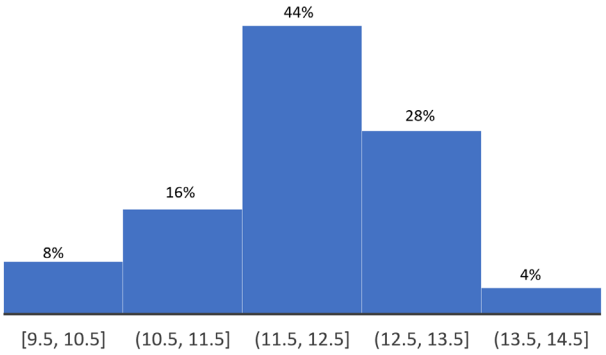

Fig. 12. Solution time histogram [seconds] for 8-subproblem class decomposition.

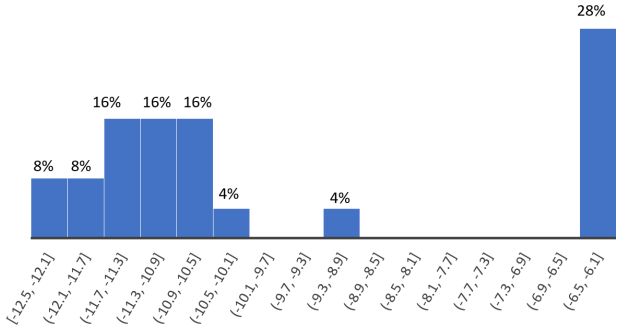

Fig. 13. Logarithmic relative error histogram for 8-subproblem class decomposition.

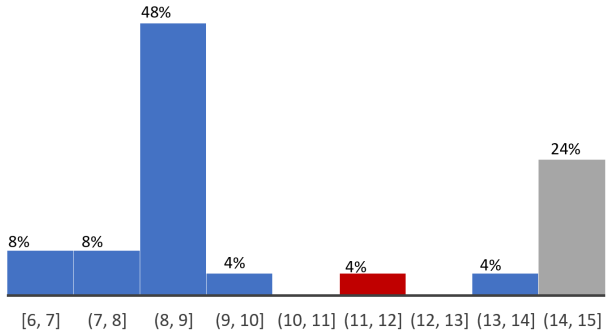

Fig. 14. Solution time histogram [seconds] for proposed learning-aided approach.

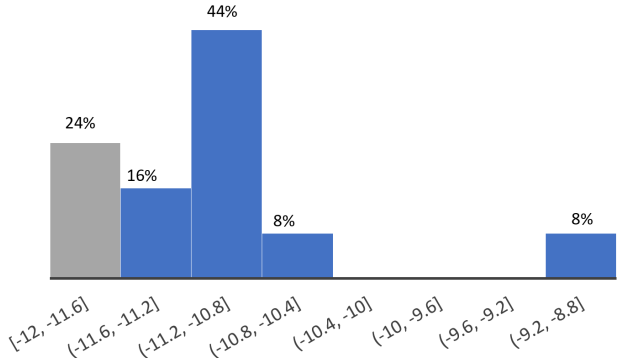

Fig. 15. Logarithmic relative error for proposed learning-aided approach.

D. Unit Commitment for IEEE 24-Bus System

The learning-aided decomposition is applied to a unit commitment problem with a considered horizon of 72 hours. We have extracted 5226 weekly load patterns from PJM [30]. The load profiles are scaled to be used for the IEEE 24-bus system. The centralized optimization problem is decomposed into all divisors of 72, representing possible classes that a load profile can belong to. The coordination strategy in [10] is used. Increasing the number of subhorizons beyond 12 increases the number of distributed optimization iterations significantly such that those classes cannot lead to the least solution time and rel. Eighty percent of the dataset is selected randomly for training and twenty percent for testing. The overall classification accuracy, precision, recall, and F_1 score metrics are above 98%, indicating the trained classifier assigns the best decomposition label to load profiles accurately.

Distributed optimization is implemented for all test load profiles with different decomposition classes. The minimum, maximum, and average relative error and average solver time for all test load profiles are reported in Table VIII. A user may conclude that having four subhorizons is the best decomposing paradigm for all load profiles. However, this table shows the average values, which is not necessarily the best choice for all load profiles. Figure 16 shows the number of samples in each decomposition class obtained by the learner. Only 24% of load profiles belong to the 4-subhorizon class, whereas 76% of samples belong to other decomposition classes. The average solution time and relative error are respectively 31% and 10^{-8} less if the proposed approach is used instead of always decomposing the problem into four subproblems.

TABLE VIII
MINIMUM, MAXIMUM AND AVERAGE RELATIVE ERROR AND AVERAGE SOLVER TIME FOR ALL LOAD PROFILES IN DATASET FOR 24-BUS SYSTEM

Number of subhorizons	max rel	min rel	Average rel	Average solver time (sec)
1	-	-	-	229
2	~0	0	~0	11
3	0.03	~0	9e-4	9
4	1e-4	0	8e-6	3
6	6e-5	0	6e-4	4
8	4e-5	0	1e-4	3.5
9	8e-5	0	2e-4	3
12	0.1	0.05	0.07	6

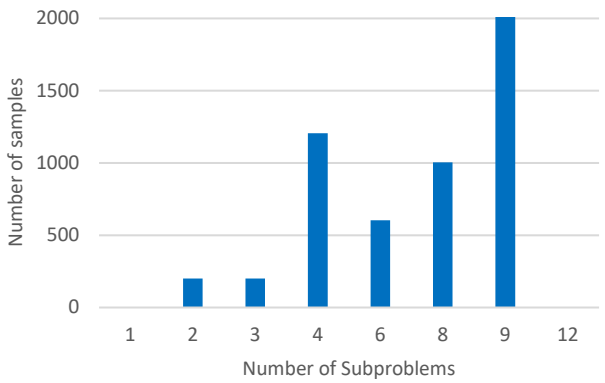

Fig. 16. Number of samples is each decomposition class using the learning-aided approach.

VI. Conclusion

In this paper, we have focused on decomposing the overall time horizon and finding the best number of subhorizons. We have observed that load values and profile patterns significantly impact the best number of subhorizons. The load is predictable and is known before solving the system scheduling problem. Possible load patterns have been decomposed into various numbers of subhorizons, and distributed optimization has been solved. Each load profile has been labeled with its corresponding best decomposition class that results in the best time-saving and relative error. We have trained a multiclass classification learner based on XGBoost, whose goal is to read the load data as the input and project it to the corresponding best decomposition class.

The simulation studies using real-world load patterns show that the proposed algorithm can efficiently and quickly find the best number of subproblems. More than 98% of the cases are predicted correctly for all studied cases. Using the best number of subproblems reduces the solution time more significantly than a naïve decomposition with a single class for all load profiles deduced from average values.

Appendix

The considered centralized SCED problem, which is a multi-interval DC OPF, is formulated as follows. The objective function is to minimize generation costs subject to generating unit limitations (a.2)-(a.5) and power flow constraints (a.6)-(a.8) under normal and contingency conditions [1].

$$\min \sum_t \sum_u a_u \cdot {p_{ut}}^2 + b_u \cdot p_{ut} + C_u \qquad (a.1)$$

s.t.

$$\underline{P}_{utc} \le p_{utc} \le \overline{P}_{utc} \qquad \forall u, \forall t, \forall c \qquad (a.2)$$

$$p_{utc} - p_{u(t-1)c} \le \mathrm{UR}_u \qquad \forall u, \forall t, \forall c \qquad (a.3)$$

$$p_{u(t-1)c} - p_{utc} \le \mathrm{DR}_u \qquad \forall u, \forall t, \forall c \qquad (a.4)$$

$$|p_{ut0} - p_{utc}| \le \Delta \qquad \forall u, \forall t, \forall c \qquad (a.5)$$

$$\underline{P}_{ij} \le \frac{\delta_{itc} - \delta_{jtc}}{X_{ij}} \le \overline{P}_{ij} \qquad \forall ij, \forall t, \forall c \qquad (a.6)$$

$$p_{uitc} - p_{ditc} = \sum_j \frac{\delta_{itc} - \delta_{jtc}}{X_{ij}} \qquad \forall i, \forall t, \forall c \qquad (a.7)$$

$$\delta_{ref,tc} = 0 \qquad \forall t, \forall c \qquad (a.8)$$

where u, t, and c are indices for generating units, time, and contingency scenarios. Subscripts i, j, and ij indicate bus i, bus j, and line ij. Parameters UR_u and DR_u refer to ramp up and down limits of unit u. δ are bus voltage angles. We refer to [11] for more details.

References

[1] A. J. Wood, B. F. Wollenberg, and G. B. Sheblé, *Power generation, operation, and control*. John Wiley & Sons, 2013.

[2] A. J. Conejo, F. J. Nogales, and F. J. Prieto, "A decomposition procedure based on approximate Newton directions," *Mathematical programming,* vol. 93, no. 3, pp. 495-515, 2002.

[3] M. H. Amini *et al.*, "Decomposition Methods for Distributed Optimal Power Flow: Panorama and Case Studies of the DC Model," in *Classical and Recent Aspects of Power System Optimization*: Elsevier, 2018, pp. 137-155.

[4] A. Kargarian *et al.*, "Toward distributed/decentralized DC optimal power flow implementation in future electric power systems," *IEEE Transactions on Smart Grid,* vol. 9, no. 4, pp. 2574-2594, 2018.

[5] D. K. Molzahn *et al.*, "A survey of distributed optimization and control algorithms for electric power systems," *IEEE Transactions on Smart Grid,* vol. 8, no. 6, pp. 2941-2962, 2017.

[6] Y. Wang, S. Wang, and L. Wu, "Distributed optimization approaches for emerging power systems operation: A review," *Electric Power Systems Research,* vol. 144, pp. 127-135, 2017.

[7] S. Kar, G. Hug, J. Mohammadi, and J. M. Moura, "Distributed State Estimation and Energy Management in Smart Grids: A Consensus ${+}$ Innovations Approach," *IEEE Journal of selected topics in signal processing,* vol. 8, no. 6, pp. 1022-1038, 2014.

[8] A. R. Malekpour and A. Pahwa, "Stochastic networked microgrid energy management with correlated wind generators," *IEEE Transactions on Power Systems,* vol. 32, no. 5, pp. 3681-3693, 2017.

[9] A. Asrari, M. Ansari, J. Khazaei, and P. Fajri, "A Market Framework for Decentralized Congestion Management in Smart Distribution Grids Considering Collaboration Among Electric Vehicle Aggregators," *IEEE Transactions on Smart Grid,* 2019.

[10] F. Safdarian, A. Mohammadi, and A. Kargarian, "Temporal Decomposition for Security-Constrained Unit Commitment," *IEEE Transactions on Power Systems,* vol. 35, no. 3, pp. 1834-1845, 2019.

[11] F. Safdarian and A. Kargarian, "Time decomposition strategy for security-constrained economic dispatch," *IET Generation, Transmission & Distribution,* vol. 13, no. 22, pp. 5129-5138, 2019.

[12] F. Safdarian and A. Kargarian, "Temporal Decomposition-Based Stochastic Economic Dispatch for Smart Grid Energy Management," *IEEE Transactions on Smart Grid,* vol. 11, no. 5, pp. 4544 - 4554, 2020.

[13] J. Guo, G. Hug, and O. K. Tonguz, "Intelligent partitioning in distributed optimization of electric power systems," *IEEE Transactions on Smart Grid,* vol. 7, no. 3, pp. 1249-1258, 2016.

[14] J. Guo, O. Tonguz, and G. Hug, "Impact of power system partitioning on the efficiency of distributed multi-step optimization," in *2017 IEEE Manchester PowerTech*, 2017: IEEE, pp. 1-6.

[15] A. Mohammadi, M. Mehrtash, A. Kargarian, and M. Barati, "Tie-line characteristics based partitioning for distributed optimization of power systems," in *2018 IEEE Power & Energy Society General Meeting (PESGM)*, 2018: IEEE, pp. 1-5.

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

[17] T. Yalcinoz and O. Köksoy, "A multiobjective optimization method to environmental economic dispatch," *International Journal of Electrical Power & Energy Systems,* vol. 29, no. 1, pp. 42-50, 2007.

[18] W. L. Snyder, H. D. Powell, and J. C. Rayburn, "Dynamic programming approach to unit commitment," *IEEE Transactions on Power Systems,* vol. 2, no. 2, pp. 339-348, 1987.

[19] V. Dieu and W. Ongsakul, "Enhanced augmented Lagrangian Hopfield network for unit commitment," *IEE proceedings-generation, transmission and distribution,* vol. 153, no. 6, pp. 624-632, 2006.

[20] S. Pineda, R. Fernández-Blanco, and J. M. Morales, "Time-Adaptive Unit Commitment," *IEEE Transactions on Power Systems,* vol. 34, no. 5, pp. 3869-3878, 2019.

[21] D. P. Bertsekas, *Nonlinear programming*. Athena Scientific, 1999.

[22] Y. Nesterov, "A method of solving a convex programming problem with convergence rate O (1/k2)," in *Soviet Mathematics Doklady*, 1983, vol. 27, no. 2, pp. 372-376.
[23] T. Goldstein, B. O'Donoghue, S. Setzer, and R. Baraniuk, "Fast alternating direction optimization methods," *SIAM Journal on Imaging Sciences,* vol. 7, no. 3, pp. 1588-1623, 2014.
[24] G. Cohen, "Auxiliary problem principle and decomposition of optimization problems," *Journal of optimization Theory and Applications,* vol. 32, no. 3, pp. 277-305, 1980.
[25] A. Liaw and M. Wiener, "Classification and regression by randomForest," *R news,* vol. 2, no. 3, pp. 18-22, 2002.
[26] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, 2016, pp. 785-794.
[27] T. Chen, T. He, M. Benesty, and V. Khotilovich, "Package 'xgboost'," *R version 0.90,* 2019.
[28] "System Data." https://sites.google.com/site/aminkargarian/test-system-data/multi-class-learning-based-temporal-decomposition-and-distributed-optimizat?authuser=0 (accessed.
[29] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in Python," *Journal of machine learning research,* vol. 12, no. Oct, pp. 2825-2830, 2011.
[30] "PJM Load Data." http://dataminer2.pjm.com/feed/load_frcstd_7_day (accessed.
[31] C. Wang, C. Deng, and S. Wang, "Imbalance-XGBoost: leveraging weighted and focal losses for binary label-imbalanced classification with XGBoost," *Pattern Recognition Letters,* vol. 136, pp. 190-197, 2020.
[32] S. Soleimani, S. R. Mousa, J. Codjoe, and M. Leitner, "A comprehensive railroad-highway grade crossing consolidation model: a machine learning approach," *Accident Analysis & Prevention,* vol. 128, pp. 65-77, 2019.

Farnaz Safdarian is a Postdoctoral Researcher at Texas A&M University. She received her Ph.D. in Electrical Engineering at Louisiana State University. She earned her B.S. and M.S. degrees in Electrical Engineering from Amirkabir University of Technology (Tehran Polytechnic) and Shahid Beheshti University, Iran, in 2011 and 2014, respectively. She has been involved in various projects and published several papers as part of her professional career. She also has work experience in the power industry as well as teaching.

Amin Kargarian (SM'20) received his Ph.D. degree in electrical and computer engineering from Mississippi State University, Starkville, MS, USA, in 2014. He was a Postdoctoral Research Associate in the Electrical and Computer Engineering Department at Carnegie Mellon University in 2014-2015. He is currently an Assistant Professor with the Electrical and Computer Engineering Department, Louisiana State University, Baton Rouge, LA, USA. His research interests include power systems optimization and machine learning.

Fouad Hasan (S'18) received his B.Sc. degree in electrical engineering (power system) from Bangladesh University of Engineering and Technology (BUET), Dhaka, Bangladesh, in 2015. He is currently pursuing his Ph.D. degree with the Department of Electrical and Computer Engineering, Louisiana State University, Baton Rouge, LA, USA. His research interests include optimization, power systems operation, electricity market, and machine learning.