5-21-2023

Learning–Assisted Constraint Filtering to Enhance Power System Optimization Performance

Fouad Hasan
Louisiana State University and Agricultural and Mechanical College

LEARNING-ASSISTED CONSTRAINT FILTERING TO ENHANCE POWER SYSTEM OPTIMIZATION PERFORMANCE

A Dissertation

Submitted to the Graduate Faculty of the
Louisiana State University and
Agricultural and Mechanical College
in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

in

The Department of Electrical and Computer Engineering

by
Fouad Hasan
B.Sc., Bangladesh University of Engineering and Technology, 2015
M.Sc., Louisiana State University and Agricultural and Mechanical College, 2022
August 2023

To my dearest mother, Rabeya Yesmin Baby

I extend my heartfelt dedication.

Amidst the struggles and the strain of

My Ph.D. journey, one of life's grand expeditions,

Her love and care

Have been my unwavering companions

Acknowledgments

I would like to express my gratitude to my Supervisor, Dr. Amin Kargarian, and all the committee members, Dr. Shahab Mehraeen, Dr. Hsiao-Chun Wu, Dr. Javad Mohammadi, Dr. Morteza Naraghi-Pour, and Dean's Representative Dr. Hongyu He.

I also express gratitude to all ECE staff and faculty members for always being kind and helpful. I would like to thank all my Louisiana State University friends who have supported me throughout my Ph.D. studies.

At last, I offer my gratitude to my mother, Rabeya Yesmin; my father, Abdul Baten; my brother, Rezaur Rahman; and my sister, Tahera Siddika. Their boundless emotional support and encouragement have been my pillar of strength throughout this prolonged journey of Doctoral studies.

Table of Contents

Acknowledgements iii

List of Tables vi

List of Figures viii

Abbreviations and Symbols x

Abstract xiii

Chapter 1. Introduction 1
1.1. Background 1
1.2. Motivation 3
1.3. Review of Machine Learning Applications to Power System Optimization 3
1.4. Contribution and Organization 7

Chapter 2. Hybrid Learning Aided Inactive Constraints Filtering Algorithm to Enhance ACOPF Solution Time 10
2.1. Introduction 10
2.2. Hybrid Regression-Classification Algorithm for Inactive Constraints Identification 11
2.3. Selecting Learning Approach and Algorithm 19
2.4. Numerical Results 22
2.5. Conclusion 30

Chapter 3. Topology-Aware Learning Assisted Branch and Ramp Constraints Screening for Dynamic Economic Dispatch 32
3.1. Introduction 32
3.2. Dynamic Economic Dispatch 33
3.3. Proposed Learning-Aided Iterative Approach 34
3.4. Numerical Simulation and Results Analysis 51
3.5. Conclusion 59

Chapter 4. Accelerating L-Shaped Two-Stage Stochastic SCUC With Learning Integrated Benders Decomposition 60
4.1. Introduction 60
4.2. Two-stage Stochastic Unit Commitment 61
4.3. Proposed Learning Assisted Benders Decomposition 65
4.4. Numerical Result 76
4.5. Conclusion 81

Chapter 5. Concluding Remark and Future Works 83
5.1. Concluding Remarks 83

5.2. Future Works 84

Appendix: Publication Information 85

References 86

Vita 91

List of Tables

2.1. Architecture of trained NNs 22

2.2. Number of total constraints and active constraints for several test systems 23

2.3. System parameters and range of variation of load 24

2.4. Input and output lengths of learners 24

2.5. Prediction accuracy measurements of the proposed algorithm for voltage constraints classification 26

2.6. Prediction accuracy measurements of the proposed algorithm for branch constraints classification 26

2.7. Cost gap of truncated OPF 29

2.8. Iteration numbers and time-saving 29

3.1. Power generation values in MW 40

3.2. Average percentage of active and pseudo-active constraints 52

3.3. Variation range 53

3.4. Hyperparameters of modified EfficientNet-B7 architecture: Added layers after flattening 53

3.5. Average FPs and FNs per scenario 54

3.6. Performance analysis: Average iteration numbers and time-saving 56

3.7. Comparison with ICG 56

3.8. Average memory requirement (MB) to build constraints and solver time for two constraints screening schemes 57

3.9. Time gain analysis using demand and DCT as learners' input 58

3.10. Hamming distance analysis 58

4.1. Hyperparameters of the learner 69

4.2. Parameter range for demand dataset generation 77

4.3. Comparison of total cuts and useful cuts.....78

4.4. Average solution time (and improvement percentage) comparison in seconds.....78

4.5. Costgap index (%) 79

List of Figures

1.1. Power system operation and optimization....2

2.1. The proposed active constraints filtering strategy. a). Active and inactive constraints. b). Proposed technique....11

2.2. Block diagram of a) regressor ($\mathcal{L}_{\Re}$) training procedure, b) classifiers ($\mathcal{L}_{\mathcal{CV}}$ and $\mathcal{L}_{\mathcal{CB}}$) training procedure and c) Utilization of trained learners....14

2.3. Root mean square (RMS) error comparison of different NN regression architecture....20

2.4. Confusion matrices for the Pegase 1354-bus system a) voltage constraints and b) branch flow constraints....27

3.1. Operating conditions known before solving the D-ED problem....34

3. 2. Block diagram of proposed classifier a) training phase and b) utilization with an embedded loop....37

3.3. Number of accumulated active constraints over iterations....40

3.4: Demand scenario generation....43

3.5. Red, green, and blue channels of DCT image corresponding to an operating condition scenario for the 118-bus system....46

3.6. Block diagram of the proposed transfer learning procedure....48

4.1. Useful cut identification for benders decomposition....61

4.2. Benders convergence using $\alpha_{\omega}^{*}, \forall \omega$ in inequality (4.2b) 67

4.3. Demand scenario generation....68

4.4. NN regressor reads demand profiles and predicts proxy variables 69

4.5. Benders lower bound improvement with cuts added cumulatively; a case of IEEE 118-bus system....70

4.6. R-Benders flowchart....71

4.7. C-Benders flowchart....75

4.8. CR-Benders flowchart....76

4.9. Average number of iterations.....78

4.10. Memory usage comparison. a. 24-bus (1h) b. 24-bus (12h) b. 118-bus (1h) d. 118-bus (3h).....80

4.11. Cumulative increment of the number of cuts for IEEE 118-bus 3h case.....81

Abbreviations and Symbols

Abbreviations

ACOPF	AC optimal power flow
CNN	Convolutional neural network
D-ED	Dynamic economic dispatch
FP	False positive
FN	False negative
ICG	Iterative constraint generation
MCBD	Multi-cut Benders decomposition
MILP	Mixed integer linear programming
ML	Machine learning
MP	Master problem
NN	Neural network
SCUC	Security-constrained economic dispatch
SP	Subproblem
TN	True negative
TP	True positive

Symbols

P_d, Q_d	Real and reactive power demand
p_{di}^L	Minimum value of load at bus i
p_{di}^U	Maximum value of load at bus i

p_g, q_g	Actual real and reactive power generation
S	Complex power
V_m	Voltage magnitude
θ_i	Voltage angle of bus i
NI_P, NI_Q	Actual net real and reactive power injection
$h_v(x)$	Set of voltage constraints
$h_l(x)$	Set of branch flow constraints
$A(\cdot)$	Set of true active constraints
$\tilde{A}(\cdot)$	Set of predicted active constraints by classifiers
$\tilde{x}$	Predicted x values by learners
Δ_{di}	Maximum perturbation range for demand at bus i
$p_{gt\omega c}$	Generator output
D_t	Nodal power demand
$f(\cdot)$	Operation cost function of thermal units
SU_g, SD_g	Startup and shutdown cost of unit g
u_{gt}	ON/OFF status of generating unit
y_{gt}	Binary variable that is equal to 1 if unit g is started up at the beginning of time period *t*, and 0 otherwise
z_{gt}	Binary variable that is equal to 1 if unit g is shut-down at the beginning of time period *t*, and 0 otherwise
J_{MP}	Master problem objective function
J_{SP}	Subproblem objective function

α_{ω}	Proxy of the subproblem objective function
$\hbar_{\beta}(\cdot)$	Accumulated cuts
$\gamma_{t\omega}, v_{gt\omega}$	Auxiliary variables
δ	Cut filtering criterion
ε	Duality gap tolerance limit
$\eta(\cdot)$	Random parameter
λ	Duality information
π_{ω}	Probability of stochastic condition
ϕ	List of filtered cuts
$\psi(\cdot)$	Numerical value of a cut

Abstract

Machine learning (ML) is a powerful tool that provides meaningful insights for operators to make fast and efficient decisions by analyzing data from power systems. ML techniques have great potential to assist in solving optimization problems within a shorter time frame and with less computational burden. AC optimal power flow (ACOPF), dynamic economic dispatch (D-ED), and security-constrained unit commitment (SCUC) are the three energy management optimization functions studied in this dissertation. ACOPF is solved every 5~15 minutes. Because of the nonconvex and complex nature of ACOPF, solving this problem for large systems is computationally expensive and time-consuming. Classification and regression learning models are used to identify inactive transmission line flow constraints and omit them from the optimization, thus relieving computational costs. D-ED is solved daily and hourly to determine the best generation schedule. A combined learning and model-based algorithm is developed to identify the status of network and thermal unit ramp-up/down constraints and formulate a reduced-size dynamic economic dispatch problem. The learners read network topology, demand, and thermal unit generation cost information as input and identify the status of network and ramp constraints as output. SCUC solved daily is a complex decision-making problem that belongs to the class of mixed-integer optimization category. Benders decomposition is an effective method for solving this class of problems but suffers from exponential worst-case computational complexity. Classification and regression learners are used to enhance the convergence performance of Benders decomposition. The learning models generate tighter cuts and filter out non-useful cuts at each Benders iteration to accelerate a two-stage stochastic SCUC problem.

Chapter 1
Introduction

1.1. Background

Optimization is a crucial component of ensuring the efficient and reliable operation of modern power systems (Fig. 1.1). The goal is to optimize the generation, transmission, and distribution of electric power while satisfying various operational constraints, such as security, reliability, and economic criteria. Traditional optimization approaches rely on computationally costly mathematical programming techniques that frequently fail to deal with the complexity and unpredictability inherent in modern power systems. Machine learning (ML) has emerged as a promising tool to address these challenges by leveraging data-driven models to optimize power system operations. In recent years, there has been a lot of interest in using ML approaches to optimize power systems. ML models can help to capture the nonlinearity, uncertainty, and dynamic nature of power system operations, which are challenging to model with traditional optimization methods. For instance, ML models can estimate electricity consumption, forecast renewable energy generation, and identify possible defects in power system components. ML can also optimize various aspects of power system operations, such as unit commitment, load scheduling, and dispatch of power plants. The benefits of using ML in power system optimization include reduced computational time, improved accuracy, and increased flexibility. However, the adoption of ML in power system optimization faces several challenges, such as data availability, model interpretability, and algorithm scalability. Machine learning techniques have shown great potential in enhancing the performance of power system optimization tasks such as optimal power flow (OPF), economic dispatch (ED), and unit commitment (UC). In OPF, machine learning algorithms can learn the complex relationships between system variables and determine the optimal setpoints for controlling the power system in real-time.

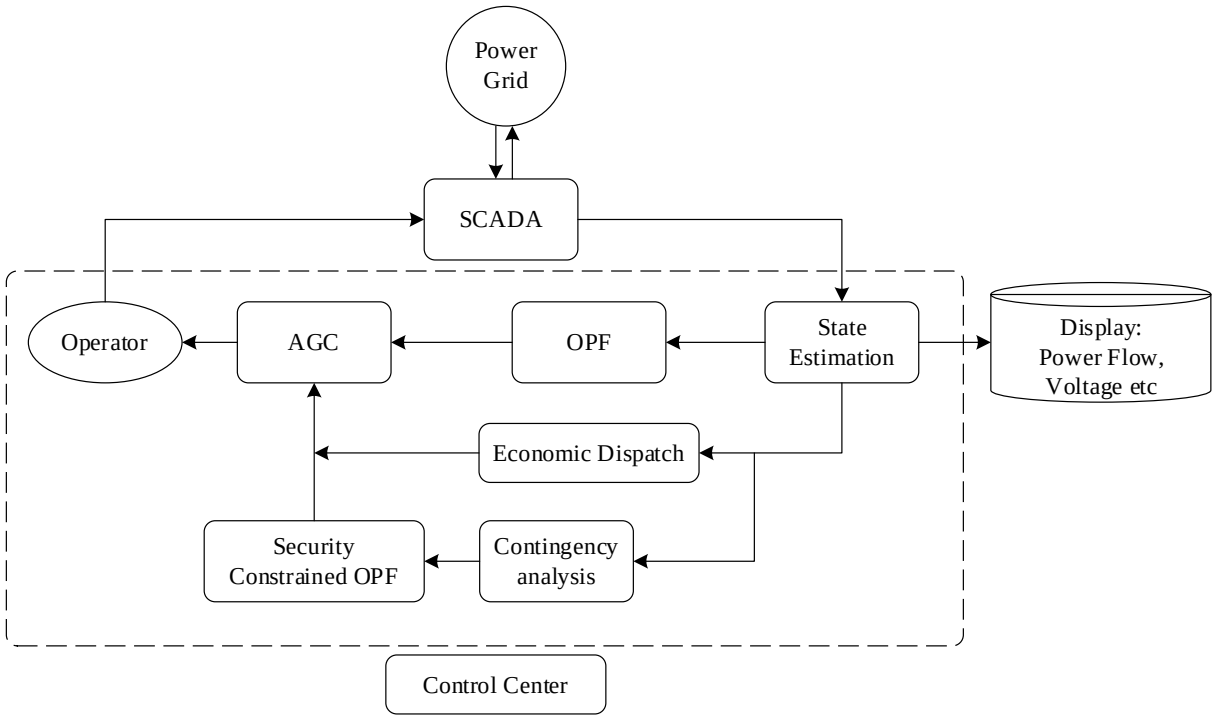


Figure 1.1. Power system operation and optimization

This can improve system efficiency and reliability and reduce operational costs. Similarly, machine learning techniques can be applied in economic dispatch to learn the relationships between power demand, generation capacity, intertemporal constraints, and other system parameters. The resulting models can then be used to optimize the dispatch of power generation units to minimize the total cost of power generation while meeting the load demand. In unit commitment, machine learning can predict the demand profile and help operators decide which generators to turn on or off at specific times to meet the predicted demand. This can help reduce the overall operational cost of the power system and ensure that the system operates reliably and safely. Overall, the application of machine learning in power system optimization is a promising approach for improving the efficiency and reliability of power systems and has the potential to lead to significant cost savings and environmental benefits.

1.2. Motivation

The application of machine learning techniques for power system optimization problems has gained traction in recent years. Despite the considerable efforts made in finding computationally efficient techniques for solving the AC optimal power flow (ACOPF), economic dispatch (ED), and unit commitment (UC) problems, the research community is still in search of reliable solutions [1-5]. The complexity of these problems has led to the adoption of various approximation, relaxation, and decomposition techniques with human engineering judgment [6-10]. However, these methods still face significant computational challenges. Machine learning techniques have emerged as a potential solution to address these issues and improve the cost-effectiveness of the solutions obtained. Recent research has focused on using machine learning models to predict active constraints, initial start values, control policies, inverter outputs, binary decision variables, and more. Using machine learning in these ways can improve the overall time gain and computational burden.

1.3 Review of Machine Learning Applications to Power System Optimization

Machine learning is a promising tool to provide a cost-effective solution to power system optimization problems and to reduce computational burden. Researchers are exploring machine learning (ML) techniques to efficiently solve and alleviate the computational complexity of NP-hard MIP problems. ML-based algorithms can be categorized into three types: end-to-end learning (with label), unsupervised/reinforcement learning (no ground truth), and algorithm-specific hybrid learning, which leverages the specific structure of the target problem to accelerate the optimization process. We review learning-based approaches to reduce the computational burden of power system optimization problems, e.g., optimal power flow, economic dispatch, and unit commitment.

The benefit of a learning-based warm start to solve ACOPF is discussed in [11]. A learner's prediction is a warm start for the optimization solver. However, this approach does not reduce the OPF size as the complete set of constraints is still used, which may not yield considerable speed improvement. Feasibility enforced deep learning is presented in [12-14] to predict generator voltage and power setpoints from demand. Lagrangian duals are combined with deep learning to enforce constraint satisfaction. In [15], deep neural networks predict active power and bus voltages. A penalty function ensures the feasibility of operational constraints. References [16-18] use black-box strategies to predict generation setpoints directly. However, learning a continuous-valued multi-dimensional variable (e.g., generation setpoints) is a demanding machine learning task. Moreover, even a slight mismatch between predicted and actual generation setpoints may yield power balance equations infeasibility or suboptimality. This makes system operators reluctant to deploy black-boxing learning-based approaches. Some recent studies combine learning and power system models to develop hybrid learning and physics-based approaches [19]. One of these approaches is to use machine learning to predict inactive/redundant optimization constraints instead of predicting the optimal output directly. Reference [20] presents experimentation on predicting transmission constraints, warm start, and affine subspace to improve mixed-integer solvers' computational performance for unit commitment. A data-driven method is developed in [21] to identify inactive and redundant constraints for single period unit commitment. A statistical learning-based ensemble control policy is presented in [22] to track real-time DCOPF. In [23], a simple neural network classifier is trained to predict DCOPF active line flow constraints. In [24], an umbrella constraint prediction algorithm is developed instead of predicting binding constraints. A constraint is an umbrella constraint if its removal changes feasible solutions of the original optimization problem. A learning-based approach is presented in [25] to classify zero

probability events and reduce joint chance constraints' computational burden for solving OPF. [26] presents a two-step prescreening approach to identify and remove non-dominating constraints. These hybrid constraint classification approaches do not consider thermal unit generation cost and power network topology alterations. Also, only spatial line flow constraints are considered, not intertemporal generating unit ramp constraints whose contribution to computational burden cannot be ignored. We aim to address these knowledge gaps in this work.

It has been more than 50 years since the development of the Benders decomposition algorithm by J. Bender (1962) [26]. The algorithm was designed to address complicating variables that, when temporarily fixed, simplify the problem significantly. It distributes the computational load between a master problem and a sub-problem (SP) based on the mathematical structure of the problem. Benders decomposition has proven to be a successful technique in a wide range of fields, including planning and scheduling, healthcare, transportation, telecommunications, energy and resource management, and chemical process design. The primary application of Benders decomposition was initially focused on solving mixed-integer linear programming (MILP) problems. Once the integer variables of a MILP problem are fixed, it can be converted into a continuous linear program (LP), which is more easily solved and can develop cuts using standard duality theory. Over time, many enhancements have been made to extend the applicability of the algorithm to a wider range of situations. As a result, the Benders decomposition technique has been widely used to solve linear, nonlinear, integer, stochastic, multi-stage, bilevel, and other optimization problems.

This section examines the current algorithmic improvements and acceleration methods available for the Benders decomposition (BD). The traditional implementation of the BD algorithm can be computationally expensive, time-consuming, and memory-intensive, with issues such as poor feasibility and optimality cuts, ineffective early iterations, and the zigzagging behavior of primal

solutions. Researchers have explored various strategies to speed up the convergence and reduce the number of iterations and time spent on each iteration. The master problem (MP) is usually solved using branch-and-bound, with the simplex approach used to solve the sub-problem. However, a significant number of cuts generated do not contribute to convergence, leading to memory occupation [27]. To address this issue, some researchers have proposed cut improvement criteria to ensure that new and useful cuts are included in the optimal solution [28]. Additionally, some researchers have observed several orders of improvement when using constraint programming to solve the MP [29]. To handle specific structures more effectively and achieve tighter constraints at the root node of the branch-and-bound tree, column generation (CG) has been introduced to MP [30, 31]. Compression, parallelization, and column generation techniques have been employed for large-scale LP sub-problems to reduce overall solution time. Adding valid inequalities to MP can significantly reduce the number of generated cuts and solution time [32]. Moreover, clustering subproblems can decrease the number of iterations required, as demonstrated in [33]. A support vector machine was employed in [34] to construct a cut classifier that identifies valuable cuts in each Benders iteration, thus reducing the size of the master problem and shortening the solution time. Another strategy proposed in [35] uses Lagrange multipliers of the Benders subproblem to aggregate optimality cuts. In [36], a cut regressor and a cut classifier were used, and five features were designed to characterize the generated cuts since there are no universal criteria to identify useful cuts [37].

1.4. Contribution and Organization

Chapter 2 presents a combined learning and model-based algorithm to predict inequality constraints' status before solving ACOPF and drop them from the optimization model to speed up the solution time. A hybrid supervised regression-classification-based approach is proposed to

identify active and inactive bus voltage and branch flow constraints of ACOPF in the learning phase. The proposed algorithm reads nodal real and reactive power demand as inputs and predicts active inequality constraints with the aim of reducing the size of OPF in a fast and efficient manner. One regression learner is trained to project generating units' production by reading demand information. The outputs of these learners are used along with demand information to train two classifiers, one for voltage constraints and another for branch flow constraints. The proposed algorithm constructs a truncated ACOPF with a subset of constraints containing sufficient information for forming the OPF feasible design space. This makes the proposed algorithm different than several existing methods that directly predict OPF results from demand. The truncated and original ACOPF problems' solutions are almost the same while solving the truncated optimization is much faster and needs less computational resources. The simulation results show the proposed algorithm's effectiveness in identifying active constraints and constructing a truncated ACOPF.

Chapter 3 presents a topology-aware learning-aided iterative algorithm to predict the necessary and sufficient branch and ramp up/down constraints information for forming the D-ED problem's feasible design space under topology alteration. The goal is to formulate a truncated D-ED with less computational cost than the original optimization problem in terms of solution time and memory usage. Demand, thermal unit generation cost, and network topology information are three inputs to the proposed algorithm. The system admittance matrix is used as an input feature for topology alterations. Using these three features, each operating condition scenario is transformed into a colorful image whose red, green, and blues channels include, respectively, demand, thermal unit generation cost, and admittance matrix information. Pre-trained convolutional neural networks are adopted, and transfer learning is used to adjust them for power system constraints

classification. The input to these classification learners is colorful images corresponding to system operating conditions, and the learners predict active and pseudo-active branch and ramp constraints. Pseudo-active constraints are inequalities that are not active at the optimal point but are required to ensure the truncated D-ED optimality and feasibility. An iterative loop is added to the proposed algorithm to find pseudo-active constraints for the learners' training phase and ensure the feasibility of truncated D-ED results if misclassifications are observed in predicting active and pseudo-active branch and ramp constraints

Chapter 4 presents a novel approach that combines learning and model-based Benders algorithm to enhance the formulation and develop a reduced Benders master problem, resulting in faster convergence. The iterative loop of the Benders decomposition technique inherently ensures both the feasibility and optimality of MIP within the tolerance limit. The key contributions of this study can be summarized as follows:

- Improved cut generation using a predicted proxy, leading to faster convergence and fewer iterations.
- Identification of the crucial cuts for creating a reduced master problem, thereby reducing memory requirements.

To expedite the data generation process, the work employs an iterative inactive constraints removal technique described in [38] to accelerate the subproblem (outlined in section III.D). Although the subproblem is linear once the integers are fixed, a significant number of inactive branch constraints can lead to a heavy computational burden in terms of time and memory requirements, particularly for large power systems. By utilizing the technique, it is possible to drastically reduce resource demands. It should be noted that this study does not introduce any

novel approaches to subproblem acceleration; however, further acceleration can be achieved by implementing the techniques described in [39].

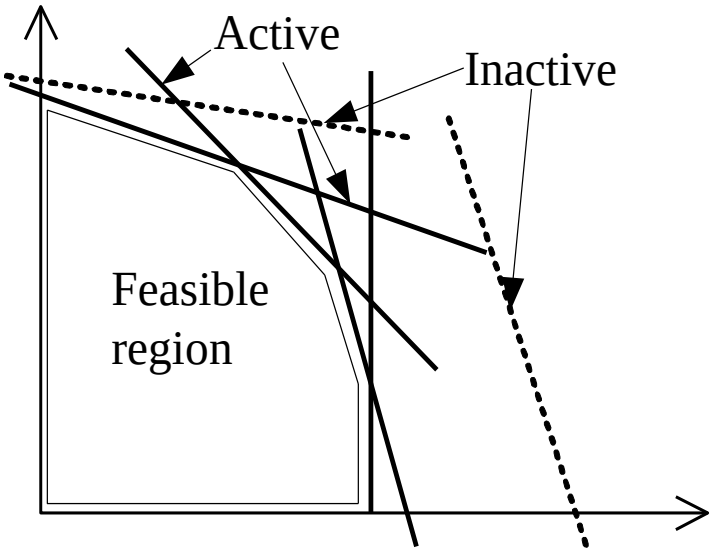
Finally, concluding remarks and suggestions for future work are provided in Chapter 5.

Chapter 2
Hybrid Learning Aided Inactive Constraints Filtering Algorithm to Enhance ACOPF Solution Time

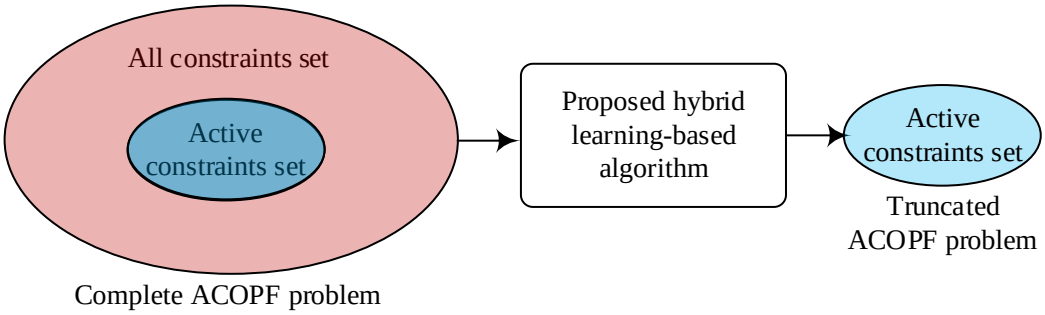
2.1. Introduction

AC optimal power flow (OPF) is one of the main energy management functions solved every 5~15 minutes. Equality and inequality constraints corresponding to power system characteristics and equipment form the OPF feasible space (also known as feasible design space, feasible region, or design space). Because of the nonconvex and complex nature of ACOPF, solving this problem for large systems is computationally expensive and time-consuming. Since the majority of OPF inequality constraints are inactive in most cases, one potential approach for relieving computational costs is to identify inactive constraints and omit them from optimization. This chapter proposes a hybrid supervised regression-classification learning-based algorithm to predict active and inactive inequality constraints before solving ACOPF solely based on nodal power demand information. The proposed algorithm is structured using a mixture of classifiers and regression learners. It predicts the required information to form the reduced OPF rather than a direct mapping of OPF results from demand. Inactive constraints are removed from the design space to construct a truncated ACOPF, as shown in Fig. 2.1. This truncated optimization problem can be solved faster than the original problem with fewer computational resources. Numerical results on several test systems show the proposed algorithm's effectiveness for predicting active and inactive constraints and constructing a truncated ACOPF.

This chapter was previously published as F. Hasan, A. Kargarian, and J. Mohammadi, "Hybrid learning aided inactive constraints filtering algorithm to enhance ACOPF solution time," *IEEE Transactions on Industry Applications,* vol. 57, no. 2, pp. 1325-1334, 2021.



(a)



(b)

Figure 2.1. The proposed active constraints filtering strategy. a). Active and inactive constraints. b). Proposed technique

2.2. Hybrid Regression-Classification Algorithm for Inactive Constraints Identification

2.2.1. Classical ACOPF Formulation

The considered ACOPF problem, presented by (2.1a)-(2.1i), is adopted from [40]. The objective function is to minimize generation costs. Nodal power balance constraints are given by (2.1b) and (2.1c). Constraints (2.1d) and (2.1e) enforce flow limits at, respectively, sending and receiving terminals of a line. The upper and lower bounds of generating units are imposed by (2.1f)

and (2.1g). Inequalities (2.1h) and (2.1i) are bus voltage magnitude and angle limits. Where g is the index of generator, l is index of line, and i, j is the index of bus, F_{max} is maximum branch flow limit.

$$\min f(p) = \sum_{g} a_g \cdot p_g^2 + b_g \cdot p_g + c_g \qquad (2.1a)$$

$s.t.$

$$g_p(\theta, V_m, p_g) = P_{bus}(\theta, V_m) + P_d - p_g = 0 \qquad (2.1b)$$

$$g_q(\theta, V_m, q_g) = Q_{bus}(\theta, V_m) + Q_d - q_g = 0 \qquad (2.1c)$$

$$h_{ls}(\theta, V_m) = |F_{ls}(\theta, V_m)| - F_{max} \leq 0 \qquad (2.1d)$$

$$h_{lr}(\theta, V_m) = |F_{lr}(\theta, V_m)| - F_{max} \leq 0 \qquad (2.1e)$$

$$p_g^{min} \leq p_g \leq p_g^{max} \qquad \forall g \qquad (2.1f)$$

$$q_g^{min} \leq q_g \leq q_g^{max} \qquad \forall g \qquad (2.1g)$$

$$V_i^{min} \leq V_i \leq V_i^{max} \qquad \forall i \qquad (2.1h)$$

$$\theta_i^{ref} \leq \theta_i \leq \theta_i^{ref} \qquad \forall i \qquad (2.1i)$$

2.2.2. Constraints Status Identification

The status of inequality constraints is unknown before solving OPF. All inequality constraints are included in the original OPF problem. The status of constraints will be known after solving the problem. If at the optimal point x^*, an inequity $h(x) \leq 0$ is satisfied as $h(x^*) = 0$, this constraint is called active or binding, otherwise inactive. To construct a truncated OPF, inactive inequality constraints should be detected and omitted from the optimization problem before solving the

problem. Detecting active and inactive constraints is cast as a binary classification problem. The constraint is labeled as 1 if $h(x^*) = 0$ and 0 otherwise.

Without loss of generality, we focus on identifying the status of bus voltage magnitude and branch flow constraints. These two sets of inequalities have high impacts on OPF computation cost. The total number of voltage magnitude and branch flow constraints is higher than other OPF inequalities, e.g., generators' upper and lower bounds. The majority of these two sets of constraints are inactive under various loading conditions. This argument is invalid for generator limits as many of these controllable devices' constraints might be active under several loading conditions.

The goal is to predict constraint status before solving OPF using only nodal demand values. For brevity, we represent branch flow constraints (2.1d) and (2.1e) and voltage magnitude constraints (2.1h) in compact forms as follows:

$$h_l(x) := \{h_{ls}(\theta, V_m); h_{lr}(\theta, V_m)\} \tag{2.2}$$

$$h_v(x) := \left\{V_i^{min} - V_i \leq 0; V_i - V_i^{max} \leq 0\right\} \tag{2.3}$$

Since the bus voltage and branch flow constraints are inherently different, we train two separate classifiers, one for bus voltage constraints and another for branch flow conditions.

2.2.3. Dataset Preparation

Before solving OPF, demand information is available. The following demand vector D is the input for learners.

$$P_d = [p_{d1}, p_{d2}, \dots, p_{dn}]^T \tag{2.4a}$$

$$Q_d = [q_{d1}, q_{d2}, \dots, q_{dn}]^T \tag{2.4b}$$

$$D = \begin{bmatrix} P_d \\ Q_d \end{bmatrix} \tag{2.4c}$$

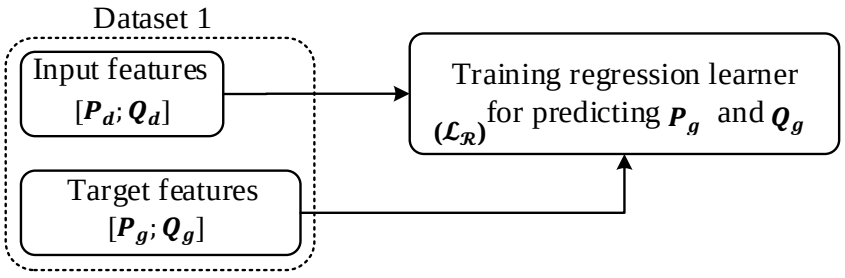
To cover possible loading situations that may occur during system operation in the training phase, we generate a set of demand scenarios as follows:

$$p_{di}^k = \left[p_{di}^L + \eta_p(k) \cdot \Delta_{di}\right] \quad \forall k \tag{2.5a}$$

$$q_{di}^k = \left[q_{di}^L + \eta_q(k) \cdot \Delta_{di}\right] \quad \forall k \tag{2.5b}$$

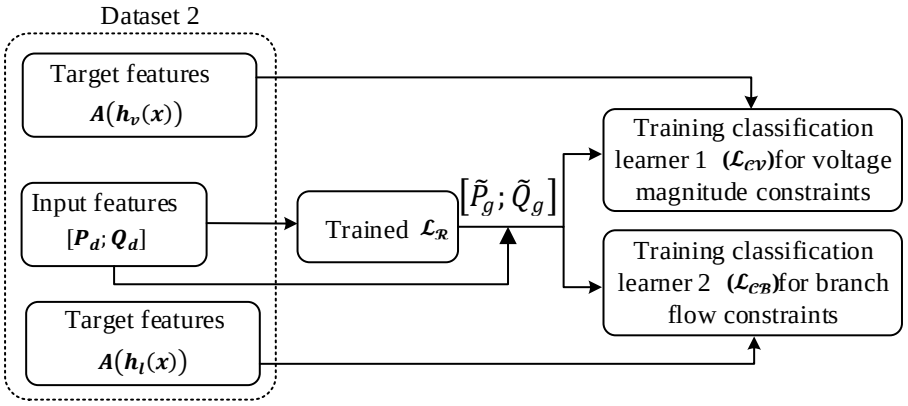
$$\Delta_{di} = p_{di}^U - p_{di}^L \tag{2.5c}$$

where $\eta_p(\cdot)$ and $\eta_q(\cdot)$ follows a uniform distribution between 0 and 1, and k is the index of demand samples. The perturbation range Δ_{di} depends on the possible minimum (P_d^L) and maximum (P_d^U) nodal demand values. For each demand scenario, OPF is solved and active and inactive bus voltage ($A(h_v(x))$) and branch flow constraints ($A(h_l(x))$) are identified and stored for training. Demand scenarios resulting in infeasible OPF are omitted. Two different nonidentical demand datasets are generated. As shown in Fig. 2.2, demand scenarios in dataset 1 are used to train a regressor ($\mathcal{L}_{\Re}$). Demand scenarios in dataset 2, along with their predicted generation values obtained from $\mathcal{L}_{\Re}$, are the training set for classifiers ($\mathcal{L}_{\mathcal{CV}}$ and $\mathcal{L}_{\mathcal{CB}}$). Using one dataset for all learners means that $\mathcal{L}_{\Re}$ is trained and then utilized with the same dataset, which is not logical.

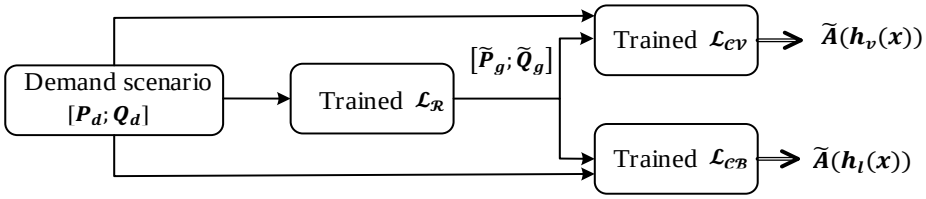


(a)

Figure 2.2. Block diagram of a) regressor ($\mathcal{L}_{\Re}$) training procedure, b) classifiers ($\mathcal{L}_{\mathcal{CV}}$ and $\mathcal{L}_{\mathcal{CB}}$) training procedure and c) Utilization of trained learners. (fig. cont'd.)



(b)



(c)

2.2.4. Proposed Training Structure

The status of voltage and branch flow constraints depends on demand values and generating units' production. Generation values are unknown before solving OPF. As shown in Fig. 2.2a, regression learner $\mathcal{L}_{\Re}$ is dedicated to predicting generation values by reading power demand. The input and target vectors of $\mathcal{L}_{\Re}$ are demand vector D and generation vector $G = [P_g; Q_g]$, $\forall g$. Several buses have neither load nor generation. These buses in nodal demand and generation vectors provide no meaningful information for the learner as their corresponding entries are always zero. Only demand buses are used to form input vector D.

By training one regressor to predict both $\tilde{P}_g$ and $\tilde{Q}_g$, the learner may capture the interaction between real and reactive powers and better understand generator dynamics. Once trained, $\mathcal{L}_{\Re}$ will predict real ($\tilde{P}_g$) and reactive ($\tilde{Q}_g$) power generated by each unit for each demand scenario. n is the number of nodes.

$$\tilde{P}_g = \left[\tilde{p}_{g1}, \tilde{p}_{g2}, \cdots, \tilde{p}_{gn}\right]^T \tag{2.6a}$$

$$\tilde{Q}_g = \left[\tilde{q}_{g1}, \tilde{q}_{g2}, \cdots, \tilde{q}_{gn}\right]^T \tag{2.6b}$$

$$\tilde{G} = \begin{bmatrix} \tilde{P}_g \\ \tilde{Q}_g \end{bmatrix} \tag{2.6c}$$

Vector D and predicted nodal power generation $\tilde{G}$ are used to form a net nodal power injection vector ($\widetilde{NI}$).

$$\widetilde{NI}_P = \left[\tilde{p}_{g1} - p_{d1}, \tilde{p}_{g2} - p_{d2}, \ldots, \tilde{p}_{gn} - p_{dn}\right]^T \tag{2.7a}$$

$$\widetilde{NI}_Q = \left[\tilde{q}_{g1} - q_{d1}, \tilde{q}_{g2} - q_{d2}, \ldots, \tilde{q}_{gn} - q_{dn}\right]^T \tag{2.7b}$$

$$\widetilde{NI} = \begin{bmatrix} \widetilde{NI}_P \\ \widetilde{NI}_Q \end{bmatrix} \tag{2.7c}$$

We train two classifiers for each system, a bus voltage constraint classifier (classifier 1 or $\mathcal{L}_{\mathcal{CV}}$ in Fig. 2.2b) and a branch constraint classifier (classifier 2 or $\mathcal{L}_{\mathcal{CB}}$ in Fig. 2b). The input vector to $\mathcal{L}_{\mathcal{CV}}$ and $\mathcal{L}_{\mathcal{CB}}$ is $\widetilde{NI}$, and their targets are $A(h_v(x))$, and $A(h_l(x))$. As shown in Fig. 2.2 (c), the classifiers read the regressor predicted generations to predict the status of constraints. The regressor predicts generation values $\tilde{G}$ for each demand scenario in dataset 2. The predicted generation values and demand scenarios in dataset 2 are used to form the net injection vector $\widetilde{NI} = \begin{bmatrix} \widetilde{NI}_P \\ \widetilde{NI}_Q \end{bmatrix} =$

$\begin{bmatrix} \tilde{P}_g - P_d \\ \tilde{Q}_g - Q_d \end{bmatrix}$ for training $\mathcal{L}_{\mathcal{CV}}$ and $\mathcal{L}_{\mathcal{CB}}$. By doing so, the classifiers are trained and utilized by predicted generation values rather than being trained with actual generation and then utilized with predicted generation values. This procedure would enhance the accuracy of $\mathcal{L}_{\mathcal{CV}}$ and $\mathcal{L}_{\mathcal{CB}}$. The pseudocode to train the learners is represented in Algorithm I.

Algorithm I Pseudocode for training learners

1. Dataset 1: Generate a set of demand scenarios $P_D(\eta_p)$ and $Q_D(\eta_q)$ by (5), and form $D_1 = \begin{bmatrix} P_d \\ Q_d \end{bmatrix}$
2. Solve original OPF for each scenario in D_1 and drop infeasible cases
3. Form $G = \begin{bmatrix} P_g \\ Q_g \end{bmatrix}$
4. Train $\mathcal{L}_{\Re}$ using D (input) and G (target) in dataset 1
5. Dataset 2: Generate a set of new $P_D(\eta_p)$ and $Q_D(\eta_q)$ by (5) and form $D_2 = \begin{bmatrix} P_d \\ Q_d \end{bmatrix}$
6. Solve original OPF for each scenario in D_2 and drop infeasible cases
7. Identify $A(h_v(x))$ and $A(h_l(x))$ corresponding to each scenario in D_2
8. Use $\mathcal{L}_{\Re}$ to predict $\tilde{G} = \begin{bmatrix} \tilde{P}_g \\ \tilde{Q}_g \end{bmatrix}$ to each scenario in D_2
9. Form $\widetilde{NI} = \begin{bmatrix} \widetilde{NI}_P \\ \widetilde{NI}_Q \end{bmatrix} = \begin{bmatrix} \tilde{P}_g - P_d \\ \tilde{Q}_g - Q_d \end{bmatrix}$
10. Train $\mathcal{L}_{\mathcal{CV}}$ using $\widetilde{NI}$ as input and $A(h_v(x))$ as target
11. Train $\mathcal{L}_{\mathcal{CB}}$ using $\widetilde{NI}$ as input and $A(h_l(x))$ as target

2.2.5. Utilization Procedure

The utilization procedure of the proposed algorithm is demonstrated in Fig. 2.2c. For a given demand, $\tilde{P}_g$ and $\tilde{Q}_g$ are determined by the trained $\mathcal{L}_{\Re}$. The given D and the predicted $\tilde{G}$ will be used to form vector $\widetilde{NI}$ that is the input of the trained $\mathcal{L}_{\mathcal{CV}}$ and $\mathcal{L}_{\mathcal{CB}}$. The output of $\mathcal{L}_{\mathcal{CV}}$ is active bus voltage constraints ($\tilde{A}(h_v(x))$), and $\mathcal{L}_{\mathcal{CB}}$ predicts active branch flow constraints ($\tilde{A}(h_l(x))$).

$\tilde{A}(h_v(x))$ and $\tilde{A}(h_l(x))$ will be used to construct a truncated optimization design space and, consequently, a truncated OPF problem as:

$$\min \sum_g a_g \cdot p_g^2 + b_g \cdot p_g + c_g \qquad (2.8a)$$

$s.t.$

$$\tilde{A}(h_v(x)) \leq 0 \qquad (2.8b)$$

$$\tilde{A}(h_l(x)) \leq 0 \qquad (2.8c)$$

$$x \in \chi$$

where χ represents all other constraints except bus voltage magnitude and branch flow constraints. Suppose all required inequality constraints are predicted correctly. In that case, the feasible space of the truncated OPF problem is the same as that of the original OPF while its size is much smaller than the original optimization problem. The pseudocode to utilize the proposed regression-classification technique to form the truncated OPF is as follows.

Proposed Utilization Algorithm

1. For a given demand vector $D = \begin{bmatrix} P_d \\ Q_d \end{bmatrix}$, run $\mathcal{L}_{\Re}$ to determine $\tilde{G} = \begin{bmatrix} \tilde{P}_g \\ \tilde{Q}_g \end{bmatrix}$
2. Form $\widetilde{NI} = \begin{bmatrix} \widetilde{NI}_P \\ \widetilde{NI}_Q \end{bmatrix} = \begin{bmatrix} \tilde{P}_g - P_d \\ \tilde{Q}_g - Q_d \end{bmatrix}$ using D and $\tilde{G}$
3. Use $\widetilde{NI}$ as input to $\mathcal{L}_{\mathcal{CV}}$ and $\mathcal{L}_{\mathcal{CB}}$ and identify $\tilde{A}(h_v(x))$ and $\tilde{A}(h_l(x))$
4. Formulate truncated ACOPF using $\tilde{A}(h_v(x))$, $\tilde{A}(h_l(x))$, and χ
5. Minimize (2.8a) subject to (2.8b), (2.8c), and χ

One may use $\mathcal{L}_{\mathcal{B}}$ of Fig. 2.2 to predict $\tilde{P}_g$ and $\tilde{Q}_g$ and then formulate and solve a modified AC power flow instead of a truncated ACOPF. Although solving AC power flow is easier than solving the truncated ACOPF, even a slight error in $\tilde{P}_g$ and $\tilde{Q}_g$ might make AC power flow results suboptimal and, more importantly, endanger power flow feasibility.

2.3. Selecting Learning Approach and Algorithm

Supervised learning approaches are selected to train $\mathcal{L}_{\mathcal{R}}$, $\mathcal{L}_{\mathcal{CV}}$, and $\mathcal{L}_{\mathcal{CB}}$ in Fig. 2.2. Various supervised machine learning approaches are available. Among them, neural networks (NNs) have shown promising performance. NNs have outperformed many other machine learning algorithms in recommendation systems, speech and image recognition, natural language processing, etc. Support vector machine (SVM) with quadratic and Gaussian functions, Gaussian process regression with exponential and quadratic kernels, and ensemble learning with bagging and boosting methods are examined for regression learners. Also, SVM with coarse quadratic and Gaussian functions, the k-nearest neighbor with coarse and weighed techniques, discriminant analysis with linear and quadratic functions, and Naïve Bayes are tested for classification learners. It is observed that while the performance of these approaches is suitable for small power systems, their performance degrades by increasing the size of the system. Also, cases are observed in which these learners failed to map a function between the input and output ACOPF training datasets (these tests and analyses are performed using MATLAB machine learning toolbox).

Neural networks are used to train $\mathcal{L}_{\mathcal{R}}$, $\mathcal{L}_{\mathcal{CV}}$, and $\mathcal{L}_{\mathcal{CB}}$ for power systems with different sizes, and promising results are obtained for the regressor and classifiers. Hence, we have selected NN for regression and constraint classification. Using activation functions, NN can effectively capture the nonlinearity and complexity of problems, such as ACOPF. A fully connected NN with mini-batch

gradient descent is used for $\mathcal{L}_{\mathcal{R}}$, $\mathcal{L}_{\mathcal{CV}}$, and $\mathcal{L}_{\mathcal{CB}}$. For $\mathcal{L}_{\mathcal{R}}$, rectified linear units (ReLU) are used in hidden layers, and linear activation functions are used for the output layer. For $\mathcal{L}_{\mathcal{CV}}$, and $\mathcal{L}_{\mathcal{CB}}$, ReLU is used in hidden layers, and the sigmoid function is used for the output. In the case of linear activation function, the output is proportional to the provided input ($X(z) = (mZ)$) whereas, based on the input, the sigmoid activation function provides the output between 0 and 1 ($X(z) = \frac{1}{1+e^{-Z}}$). The derivative of the activation function is used in the error backpropagation algorithm, which is a process to optimize each neuron's weight. The loss function of $\mathcal{L}_{\mathcal{R}}$ is the mean squared error (MSE).

$$MSE = \frac{\sum_{k=1}^{K}\left(X^k - \tilde{X}^k\right)^2}{K} \tag{2.9}$$

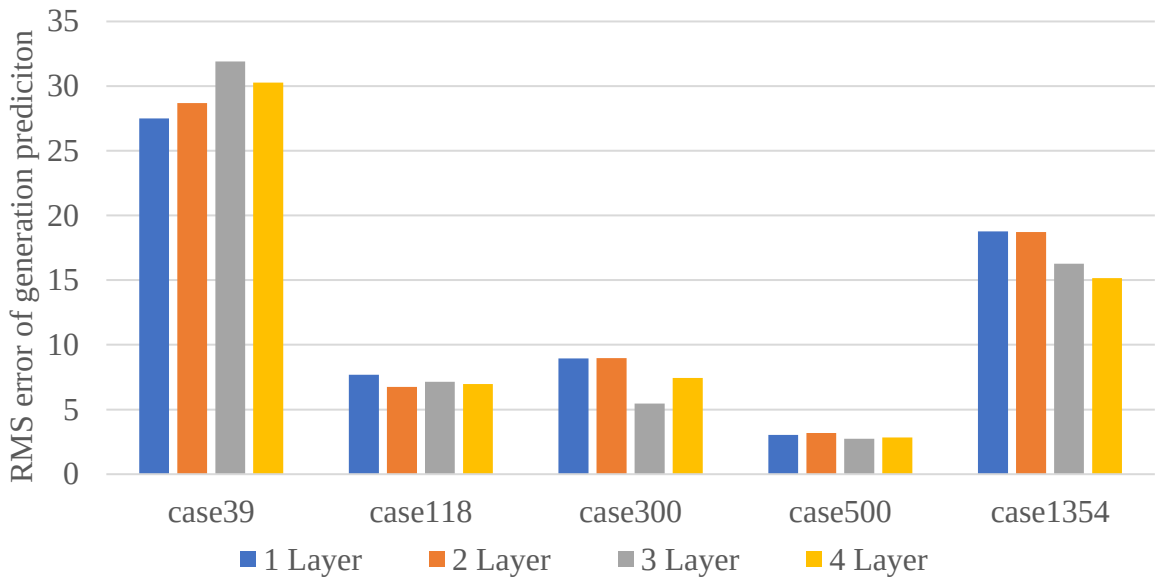


Figure 2.3. Root mean square (RMS) error comparison of different NN regression architectures.

Real and reactive powers are used to update weights ($\frac{\sum_{k=1}^{K}[(P_g^K - \tilde{P}_g^K)^2 + (Q_g^K - \tilde{Q}_g^K)]}{K}$). Adam optimizer is used to find the optimal weight values and train the learners. Various architectures are tested

with different numbers of layers, epochs, and batch sizes. Fig. 2.3 illustrates the results obtained by regressors with different numbers of hidden layers. Merely increasing the number of layers does not improve the prediction accuracy for all systems. One hidden layer is selected for minimalistic learners. Table 2.1 depicts the learners' architecture and hyperparameters used in this work. Although we have obtained promising results with these simple architectures, one can use more complex architectures to obtain better results.

2.3.1. Classifier Loss Function

Training a neural network is based on solving an optimization problem to find the best weights of a loss function. The conventional loss function is binary cross-entropy for a typical binary classification problem. Although this function works well for many problems, it may not show good performance for imbalanced datasets [41]. Since most voltage and line flow constraints are inactive, the percentage of inactive constraints in dataset 2 is much higher than that of active constraints. Due to this imbalance, the model tends to be overfitted to the class with a higher percentage in the dataset, i.e., inactive class.

To avoid this bias, F_β-score is used as the classification loss function. Hyperparameter β controls the importance of precision and recall. Maximizing recall minimizes the number of false negatives (FNs), and maximizing precision reduces false positives (FPs). The objective is to improve recall without hurting precision, which is conflicting. As reliability is critical for power, reducing the percentage of FN is desirable. We set β=2.

$$F_\beta = (1+\beta^2)\frac{Precision \cdot Recall}{\beta^2 \cdot Precision + Recall} = \frac{(1+\beta^2)\cdot TP}{(1+\beta^2)\cdot TP + FP + \beta^2 \cdot FN} \quad (2.10)$$

where

$$\text{Precision / Positive predictive value (PPV)} = \frac{TP}{FP+TP} \quad (2.11a)$$

$$\text{Recall / True positive rate (TPR)} = \frac{TP}{TP+FN}. \quad (2.11b)$$

2.3.2. Data Scaling

Data scaling, e.g., normalization and standardization, is a preprocessing step before training the learners. This step improves the numerical stability of calculations and enhances the prediction accuracy. The data are normalized by (2.12).

$$X_{noralized} = \frac{X - X^{min}}{X^{max} - X^{min}} \quad (2.12)$$

Table 2.1. Architecture of trained NNs

Learner	Training parameters	Activation function	Loss function	Optimizer
Regressor $\mathcal{L}_{\mathcal{R}}$	Hidden layer=1, Neuron=256, Epochs=1000 (with early stopping and patience =100), Batch size=100, Validation split=20%	ReLU & Linear	MSE	Adam
Classifier $\mathcal{L}_{\mathcal{CV}}$, $\mathcal{L}_{\mathcal{CB}}$	Hidden layer=1, Neuron=256, Epochs=1000 (with early stopping and patience =100), Batch size=100, Validation split=20%	ReLU & Sigmoid	F2 Loss	Adam

2.4. Numerical Results

The proposed algorithm's effectiveness for detecting active and inactive constraints is tested on several small, medium, and large systems. Test systems are adopted from the standard PGLib-OPF benchmark library [42]. MATPOWER interior point solver is used to solve OPF [40]. Python (v3.7.3) based Keras framework (v2.3.1) is used with TensorFlow to train learners. Simulations are carried out on a personal computer with a 3.70 GHz Intel(R) Xeon(R) CPU, eight cores, and

16 GB of RAM. We have posted our code on GitHub and have uploaded the data used in numerical studies to IEEE DataPort as an open access dataset (DOI: 10.21227/kege-qv50).

2.4.1. Average Number of Active and Inactive Constraints

Table 2.2 shows the number of voltage and branch constraints for the original OPF and truncated OPF problems. The second column shows the total number of voltage and branch constraints, and the third column depicts the average number of active voltage and branch constraints under various loading conditions.

Table 2.2. Number of total constraints and active constraints for several test systems

System	Original OPF (Voltage, Branch flow)	Truncated OPF (Active Voltage, Branch flow)	Inactive, Active
case39_epri	78, 92	5, 2	96%, 4 %
case118_ieee	236, 372	12, 2	99%, 1 %
case300_ieee	600,822	32,3	97.5%,2.5%
case500_tamu	1000,1192	16,4	99%,1%
case1354_pegase	2708, 3982	50, 20	99%, 1%

For the 39-bus system, for instance, the total number of voltage and branch constraints are 78 and 92, respectively, out of which, on average, five voltage constraints and two branch flow constraints are active. It is observed that larger systems have a higher percentage of inactive constraints. This shows the potential advantage of detecting active constraints to construct a truncated OPF problem instead of the original OPF. For the 39-bus and 118-bus systems, for instance, the number of constraints of the truncated OPF problem is, on average, 55% (including all equality and inequality constraints of (2.1)) less than that of the original OPF.

2.4.2. Inactive Constraints Identification by Proposed Hybrid Algorithm

Training: Nodal power demand is varied using uniform random distribution to generate possible demand scenarios over a long operation horizon. Table 2.3 shows the load perturbation range (Δ_d)

as compared to MATPOWER baseload. The range is obtained by monotonically decreasing and increasing the base-case load until the simulation fails to converge. This range is narrower for the larger systems. OPF is solved for each demand scenario. One regression learner is trained for each system. As shown in Table 2.4, the length of the output of $\mathcal{L}_{\Re}$ is equal to twice the number of generators, whereas the length of the output of $\mathcal{L}_{\mathcal{CB}}$ ($\mathcal{L}_{\mathcal{CV}}$) is equal to the number of branches (buses). Active voltage and branch flow constraints are labeled '1', and inactive constraints are labeled '0' during dataset preparation. Two classifiers are trained for each test system. We have used the same architecture for all learners for ease of replication of simulations and to show the proposed algorithm performance with simple machine learning architectures.

Table 2.3. System parameters and range of variation of load

System	NB/NL/NG	Δ_d	No of Scenario		
			Regressor (Dataset1)	Classifier (Dataset2)	Testing
case39	39/46/10	70% to 130%	2000	2000	882
case118	118/186/54	70% to 130%	2000	2000	2000
case300	300/411/69	92% to 104%	2000	2000	1641
case500	500/597/90	70% to 109%	2000	2000	3000
case1354	1354/1991/260	70% to 110%	1500	1500	1200

* NB/NL/NG stands for number of Node, Branch, and Generator, respectively

Table 2.4. Input and output lengths of learners

System	Regressor ($\mathcal{L}_{\Re}$)		Classifiers ($\mathcal{L}_{\mathcal{CV}}$, $\mathcal{L}_{\mathcal{CB}}$)		
	D	$P_g; Q_g$	NI	h_v	h_l
case39_epri	42	10*2	78	39	46
case118_ieee	189	54*2	236	118	186
case300_ieee	374	69*2	600	300	411
case500_tamu	400	90*2	1000	500	597
case1354_pegase	1332	260*2	2708	1354	1991

2.4.3. Testing

The size of training and test datasets for each studied system is provided in Table 2.3. We use different train-test split ratios. Common ratios are 80%-20%, 70%-30%, and 50%-50%. We have used split ratios with more test scenarios to validate trained learners under various loading conditions. The original OPF problem is solved for each test scenario to determine the actual active/inactive constraints. The proposed hybrid algorithm is also applied to predict active/inactive constraints. Four primary indices are introduced to interpret predicted results and analyze the accuracy of the proposed algorithm.

- ***True positives*** (TP) are cases in which a constraint is predicted to be ACTIVE, and its actual status is also ACTIVE.
- ***True negatives*** (TN) are cases where the prediction is INACTIVE, and the actual output is INACTIVE.
- ***False positives*** (FP) are cases where the prediction is ACTIVE, but the actual output is INACTIVE (type I error).
- ***False negatives*** (FN) are cases where the prediction is INACTIVE, but the actual output is ACTIVE (type II error).

In addition, we use the following statistical metrics to analyze the quality of the truncated OPF in detail.

$$\text{Accuracy}= \frac{TP+TN}{TP+TN+FP+FN} \tag{2.13a}$$

$$\text{Misclassification}= \frac{FP+FN}{TP+TN+FP+FN} \tag{2.13b}$$

$$\text{False negative rate (FNR)}=1-TPR \tag{2.13c}$$

True negative rate (TNR)/specificity = $\frac{TN}{TN+FP}$ $(2.13d)$

False positive rate (FPR)=1-TNR $(2.13e)$

False discovery rate (FDR)=1-PPV $(2.13f)$

Negative predictive value, NPV= $\frac{TN}{FN+TN}$ $(2.13g)$

False omission rate (FOR)=1-NPV $(2.13h)$

Table 2.5. Prediction accuracy measurements of the proposed algorithm for voltage constraints classification

systems	FN	FP	TN	TP	NPV	PPV	TPR	TNR	Misclassification	Accuracy
case39_epri	0.01%	4.4%	88%	7.2%	99.8%	61.6 %	98.3%	95.1%	4.6%	95.4%
case118_ieee	0.001 %	1.0%	9.2%	4.6%	99.8%	80.8 %	97.7%	98.8%	1.2%	98.8%
case300_ieee	0.05%	1.7%	92.9%	5.3%	99.9%	75.8 %	99.0%	98.2%	1.8%	98.2%
case500_tamu	0.02%	1.1%	97.2%	1.5%	99.9%	57.2 %	98.8%	98.8%	1.2%	98.8%
case1354_pegase	0.01%	1.2%	97.9%	0.84%	99.9%	41.1 %	97.9%	98.8%	1.2%	98.8%

Table 2.6. Prediction accuracy measurements of the proposed algorithm for branch constraints classification

systems	FN	FP	TN	TP	NPV	PPV	TPR	TNR	Misclassification	Accuracy
case39_epri	0%	0.03 %	97.4%	2.5%	99.9%	98.5 %	99.9%	99.9%	0.1%	99.9%
case118_ieee	0.02%	0.09 %	99.0%	0.84%	99.9%	90.1 %	96.8%	99.9%	0.1%	99.9%
case300_ieee	0%	0.04 %	99.3%	0.65%	100%	94.3 %	100%	99.9%	0.1%	99.9%
case500_tamu	0%	0.58 %	98.9%	0.41%	99.9%	41.7 %	98.8%	99.4%	0.6%	99.4%
case1354_pegase	0%	3.1%	96.4%	0.53%	100%	14.7 %	100%	96.9%	3.1%	96.9%

Tables 2.5 and 2.6 show these indices for several test systems. We have selected the Pegase 1354-bus test system and constructed a confusion matrix shown in Fig. 2.4. Each test scenario contains 2708 upper/lower bus voltage magnitude constraints and 3982 sending/receiving branch flow

	Actual inactive	Actual active	
Predicted inactive	3182349 97.9% True negative	588 0.01% False negative	NPV=99.9% FOR=0.1%
Predicted active	39280 1.2% False positive	27383 0.84% True positive	PPV=41.1% FDR=58.9%
	TNR=98.8% FPR=01.2%	TPR=97.9% FNR=2.1%	Accuracy = 98.8% Misclassification =1.2 %

(a)

	Actual inactive	Actual active	
Predicted inactive	3132262 96.4% True negative	0 0% False negative	NPV=100% FOR=0.0%
Predicted active	100024 3.1% False positive	17314 0.53% True positive	PPV=14.7%% FDR=85.3%
	TNR=96.9% FPR=3.1%	TPR=100% FNR=0%	Accuracy=96.9% Misclassification =3.1%

(b)

Figure 2.4. Confusion matrices for the Pegase 1354-bus system a) voltage constraints and b) branch flow constraints.

limits. Hence, for 1200 test scenarios, the actual and predicted status of 2708×1200 voltage and 3982×1200 branch flow constraints are observed to calculate the indices shown in Fig. 2.4. Green blocks in the second column of Figs. 2.4a and 2.4b show that 97.9% of bus voltage constraints and 96.4% of branch constraints are true negatives, which means they are correctly predicted to be inactive. In the third column, green blocks depict that 0.84% and 0.53% of voltage and branch constraints are true positives, which means they are correctly predicted to be active. As shown in orange blocks in the second column, 1.2% of voltage constraints and 3.1% of branch constraints are misclassified to be active. This is the type I error (false positives), meaning these actual inactive constraints are predicted to be active and included in the truncated OPF. This is not critical as these

few constraints do not change the truncated feasible space (i.e., do not change the OPF solution) and have no considerable impact on the computational burden of the truncated OPF. The type II error (false negative), meaning actual active constraints are predicted to be inactive, is undesirable.

As shown in orange blocks in the third column of the confusion matrices, the type II error is very close to zero percent. A simple iterative loop is enough to capture these FNs. TPR for voltage and branch flow constraints is 97.9% and 100%, showing that roughly most actual active constraints are predicted to be active. TNR pertaining to voltage and branch constraints is 98.8% and 96.9%, respectively, showing the percentage of actual inactive constraints predicted to be inactive. NPV for both voltage and branch flow constraints is roughly 100% showing that most predicted inactive constraints are truly inactive. The misclassified constraints are mainly FP, meaning no important information is lost from the feasible space of the truncated OPF. Therefore, the solution of the constructed truncated OPF will be similar to that of the complete OPF formulation.

Tables 2.5 and 2.6 show that the FN index for all cases is negligible. We have observed a few misclassified voltage constraints. A detailed analysis reveals that these constraints are not heavily binding and have a small impact on the truncated feasible space. That is, including or omitting these constraints from the truncated OPF changes the optimal solution very slightly. Although no FN misclassification is observed for most of the studied cases, the solution of truncated OPF is not always guaranteed to match that of the original OPF. In such cases with nonzero FN, the solution of truncated OPF might be infeasible for the original OPF. An iterative constraints inclusion technique can be used along with the predicted constraints to ensure the guaranteed feasibility of the solution. The average cost gap is used as an index to measure how close the truncated and original OPF solutions are.

$$\text{Cost Gap\%} = \frac{|f^{T-OPF} - f^{OPF}|}{f^{OPF}} \times 100 \tag{2.14}$$

The smaller the index is, the more accurate the solution of the truncated OPF will be. The values reported in Table 2.7 show that the truncated OPF (T-OPF) solution is very close to that of the original OPF.

Table 2.7. Cost gap of truncated OPF

System	Cost gap
case39_epri	4e-06%
case118_ieee	3e-07%
case300_ieee	5.9e-05
case500_tamu	7.5e-07 %
case1354_pegase	3e-05%

Table 2.8. Iteration numbers and time-saving

Systems	Solution time (s) (Total/avg)		Number of iterations (Avg)		Time saving
	OPF	T-OPF	OPF	T-OPF	
case39_epri	53/0.06	35/0.04	16	14	33%
case118_ieee	200/0.10	140/0.07	15	14	30%
case300_ieee	583/0.355	361/0.22	35	25	38%
case500_tamu	1170/0.39	750/0.25	25	18	35%
case1354_pegase	2640/2.20	1800/1.50	42	38	32%

Table 2.8 shows the number of iterations of the interior point method and computation time. OPF is solved for all test scenarios (the number of scenarios is given in Table 2.3). The total runtime, the average runtime, and the average number of iterations per scenario are reported. The time for learners to identify active/inactive constraints is comparatively much lower than OPF runtime and is neglected. The time-saving values are in comparison with the original OPF. The number of iterations does not decrease significantly. However, since the number of function

evaluations per iteration reduces by omitting inactive constraints, the solution time per iteration and the total time decrease. The average time of each iteration can be calculated by dividing the total time by the number of iterations. For instance, for the IEEE 118-bus system, the average time of each iteration of the interior point method decreases from 6.7ms for the original OPF to 5ms for the truncated OPF, a 30% time-saving.

In summary, Tables 2.7 and 2.8 show the promising advantage of the proposed algorithm for reducing the ACOPF problem's computation time while providing a highly accurate solution.

2.5. Conclusion

This work presents a hybrid regression-classification algorithm to identify the active and inactive voltage and branch flow constraints for OPF before solving the optimization problem. It is observed that most voltage and branch flow constraints are inactive, even if the system load changes, and have no impact on the OPF solution. The proposed learning algorithm identifies inactive inequality constraints and creates a truncated OPF problem. The proposed algorithm reduces the size of the OPF problem and its computation costs. The simulation studies show that the proposed algorithm can efficiently and quickly separate active and inactive bus voltage and branch flow constraints based on reading the predicted nodal real and reactive power demand. The results show that more than 99% of voltage and branch constraints are predicted correctly, and omitting them saves time for solving ACOPF. Further analysis of the small fraction (less than 1%) of misclassified constraints shows they are not heavily binding. Their corresponding impact on the OPF feasible space is negligible and thus affects the OPF solution very slightly.

We have tested several learning algorithms, generated diverse samples to ensure that the learners observe various patterns in the training phase, and trained learners with different

hyperparameters to obtain high-quality results with a low false-negative percentage. Another reason for the low false negative percentage is the small number of active constraints in power systems optimization problems.

Advanced approaches, such as generative adversarial networks [43-45], can be used to produce more realistic operating scenarios to form a training database. Also, other constraints such as transformers constraints, phase shifter constraints, load shedding constraints, power electronic converter constraints, capacitor banks, FACTS devices, and battery storage constraints can be included in OPF, and classifiers can be used to identify inactive constraints and drop them from the optimization formulation. In addition, the proposed algorithm can be applied to other power system scheduling problems, such as unit commitment, to reduce their computational burden.

A research direction is to investigate strategies for penalizing false-negative classes in learners' objective functions to reduce the possibility of misclassification of true active constraints. This would be useful for problems with a high percentage of active constraints as compared to total constraints. Another research direction is to develop combined learning techniques and system models to consider grid topology and generation cost changes in active/inactive constraints prediction. This direction is suitable for the application of the proposed algorithm on electricity market problems. In addition, to enhance the solution speed for DCOPF, one may identify the status of all inequality constraints and then solve the first-order optimality conditions based on the system of linear equations instead of solving a truncated DCOPF using optimization techniques. Predicting the sets of active and inactive constraints in the presence of uncertainties, such as renewable sources, is another interesting research path.

Chapter 3
Topology-Aware Learning Assisted Branch and Ramp Constraints Screening for Dynamic Economic Dispatch

3.1. Introduction

Multi-interval or dynamic economic dispatch (D-ED) is the core of various power system management functions. This optimization problem contains many constraints, a small subset of which is sufficient to enclose the D-ED feasible region. This paper presents a topology-aware learning-aided iterative constraint screening algorithm to identify the feasibility outlining subset of the network and generating units ramp up/down constraints and creating a truncated D-ED problem. We create a colorful image from nodal demand, thermal unit generation cost, and network topology information. Convolutional neural networks are trained for constraint status identification using colorful images corresponding to system operating conditions and transfer learning. Filtering inactive line flow and ramp up/down constraints reduces optimization's size and computational burden, resulting in a reduction in solution time and memory usage. Dropping all inactive branch and ramp constraints may activate some of these originally inactive constraints upon solving the truncated D-ED. A loop is added to form a constraints coefficient matrix iteratively during training dataset preparation and algorithm utilization. This iterative loop guarantees truncated D-ED results feasibility and optimality. Numerical results show the proposed algorithm's effectiveness in constraint status prediction and reducing the size and solution time of D-ED. Multi-time interval economic dispatch, also known as dynamic economic dispatch (D-ED), is solved daily and hourly for many energy management functions in power systems. D-ED has many constraints, including transmission network constraints and generating unit ramp limitations

This chapter was previously published in F. Hasan and A. Kargarian, "Topology-Aware Learning Assisted Branch and Ramp Constraints Screening for Dynamic Economic Dispatch," *IEEE Transactions on Power Systems,* vol. 37, no. 5, pp. 3495-3505, 2022.

[46]. The curse of dimensionality and computational cost increase with the system size and scheduling time intervals. Despite improvements in solvers' performance, processing technology, and computing memory, D-ED's solution time and resource prerequisites continue to be crucial factors.

3.2. Dynamic economic dispatch

The considered problem is a multi-interval economic dispatch with generating unit ramp up and down constraints. The objective function is to minimize generation costs subject to power balance (3.1b), generation limits (3.1c), generating units ramp up and down limitations (3.1d) and (3.1e) denoted by $\hbar_{\mathcal{RU}}(x)$ and $\hbar_{\mathcal{RD}}(x)$, and transmission line flow limits (3.1f) denoted by $\hbar_{\mathcal{L}}(x)$.

$$\min_{p} \sum_{t} \sum_{u} \gamma_{ut} p_{ut} \tag{3.1a}$$

s.t.

$$\sum_{u} p_{ut} = \sum_{n} d_{nt} \qquad \forall t \tag{3.1b}$$

$$P_u^{min} \leq p_{ut} \leq P_u^{max} \qquad \forall u, \forall t \tag{3.1c}$$

$$\hbar_{\mathcal{RU}}(x): p_{ut} - p_{ut-1} \leq \mathcal{RU}_u \qquad \forall u, \forall t \tag{3.1d}$$

$$\hbar_{\mathcal{RD}}(x): p_{ut-1} - p_{ut} \leq \mathcal{RD}_u \qquad \forall u, \forall t \tag{3.1e}$$

$$\hbar_{\mathcal{L}}(x): -P_l^{max} \leq \boldsymbol{SF}_l\left(\boldsymbol{p}_t^{inj} - \boldsymbol{D}_t\right) \leq P_l^{max} \quad \forall l, \forall t \tag{3.1f}$$

where t, u, n, and l are indices for time, units, buses, and lines, respectively. Variable p_{ut} is power produced by unit u at time t. Parameter γ_{ut} denotes generation cost. Parameters $\mathcal{RU}_u$ and $\mathcal{RD}_u$

are ramp up and down limits of generating unit u. $\boldsymbol{SF}$ is the generation shift factor matrix. Parameter d_{nt} is demand at bus n at time t. Bus injection and demand matrices are $\boldsymbol{p}_t^{inj}$ and $\boldsymbol{D}_t$.

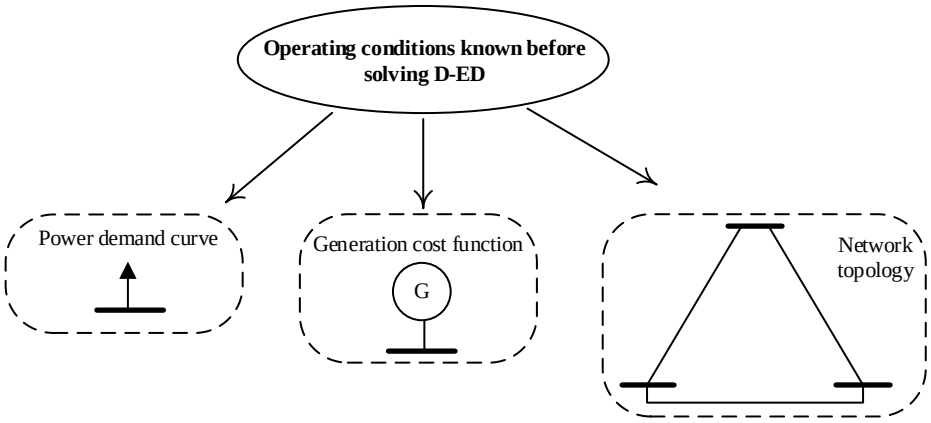


Figure 3.1. Operating conditions known before solving the D-ED problem

3.3. Proposed learning aided iterative approach

The proposed learning-aided truncated economic dispatch approach is presented in this section.

3.3.1. Truncated D-ED

Branch flow and generating unit ramp up and down constraints constitute a large portion of the D-ED constraint set. Branch flows make constraints geographically dependent, and ramp up/down limitations introduce intertemporal dependency. These constraints, particularly branch constraints, contribute significantly to computational burden and memory usage, yet most have no impact on the optimization feasible design space. The status of these constraints depends on system operating conditions shown in Fig. 3.1, including power demand, thermal unit generation cost, and network topology that are known before solving D-ED. The demand varies more significantly than the other two features. However, depending on thermal unit generation costs and network topology, active constraints may differ for a given demand value. Also, while demand and electricity market prices

are correlated, demand might not be correlated to generation costs. Thus, these three features are selected to identify the status constraints.

Fig. 3.2a and 3.2b show, respectively, an overview of the proposed constraint classification training algorithm and its utilization. The objective of the proposed algorithm is to reformulate (3.1) by the following truncated D-ED.

$$\min_{p} \sum_{t} \sum_{u} \gamma_{ut} p_{ut} \tag{3.2a}$$

s.t.

$$(1b)\ \&\ (1c) \tag{3.2b}$$

$$\tilde{\mathcal{A}}(\hbar_{\mathcal{RU}}(x)) \leq 0 \tag{3.2c}$$

$$\tilde{\mathcal{A}}(\hbar_{\mathcal{RD}}(x)) \leq 0 \tag{3.2d}$$

$$\tilde{\mathcal{A}}(\hbar_{\mathcal{L}}(x)) \leq 0 \tag{3.2e}$$

where $\tilde{\mathcal{A}}(\hbar_{\mathcal{RU}}(x))$, $\tilde{\mathcal{A}}(\hbar_{\mathcal{RD}}(x))$, and $\tilde{\mathcal{A}}(\hbar_{\mathcal{L}}(x))$ denote sets of ramp up, ramp down, and line flow constraints required to ensure that the truncated optimization problem (3.2) is equivalent to (3.1). The proposed learning-aided algorithm is designed based on the following two remarks.

3.3.1.1. Remark 1

For a given network topology, demand, and generation cost scenario, if constraints (3.1d) – (3.1f) are satisfied with equality, they must be included in the optimization problem.

$$p_{u,t} - p_{u,t-1} = \mathcal{RU}_u \qquad \forall u, t \in \Omega^t_{\mathcal{RU}} \qquad (3.3)$$

$$p_{u,t-1} - p_{u,t} = \mathcal{RD}_u \qquad \forall u, t \in \Omega^t_{\mathcal{RD}} \qquad (3.4)$$

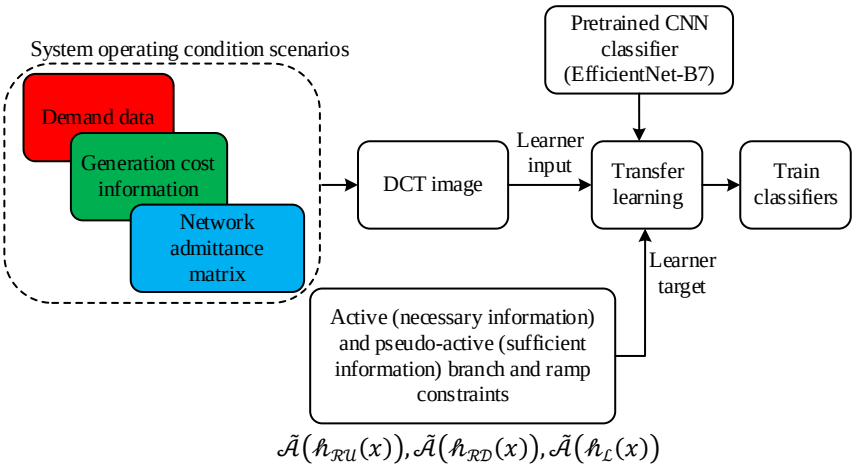
$$\left|\boldsymbol{SF}_l\left(\boldsymbol{p}^{inj}_t - \boldsymbol{D}_t\right)\right| = P^{max}_l \qquad \forall l, t \in \Omega^t_{\mathcal{L}} \qquad (3.5)$$

where $\Omega^t_{\mathcal{RU}}$ denote the set of generators with active ramp up limits at time t, $\Omega^t_{\mathcal{RD}}$ indicates the set of generators with active ramp down at time t, and $\Omega^t_{\mathcal{L}}$ is the set of active line constraints at time t.

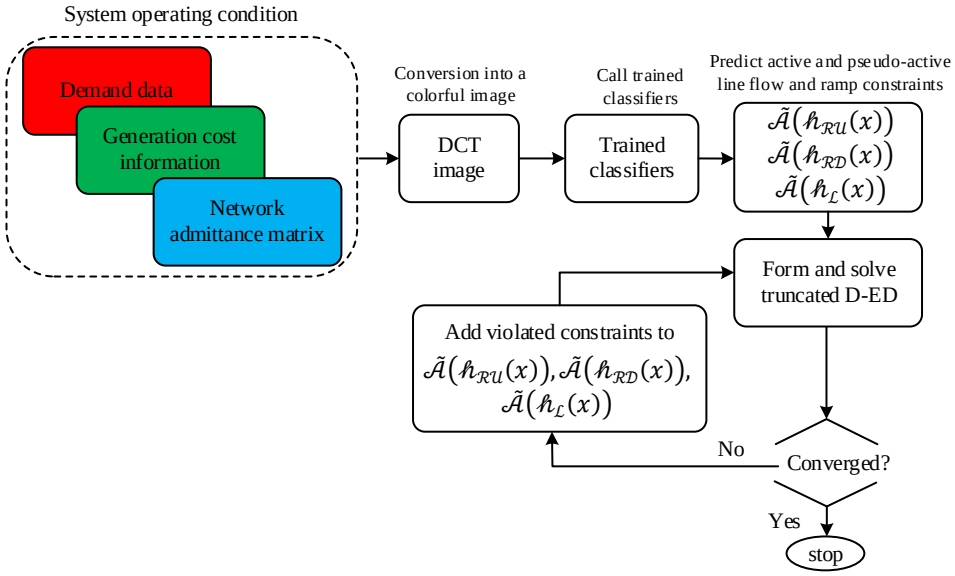
3.3.1.2. Remark 2

If generation cost coefficients of several thermal units are the same, multiple non-unique optimal solutions with different generation schedules but the same objective value might exist. Thus, dropping all inactive line flow, ramp up, and ramp down constraints and keeping only active constraints obtained at a given optimal solution, and resolving the truncated problem may trigger several originally inactive constraints to be activated. Removing these particular non-binding constraints, named *pseudo-active constraints* in this paper, may move the optimal solution to a point with the same objective value as the original problem but with a different generation schedule that is infeasible from the original problem's perspective. Thus, (3.3)–(3.5) are necessary but not sufficient to form $\tilde{\mathcal{A}}(\hbar_{\mathcal{RU}}(x))$, $\tilde{\mathcal{A}}(\hbar_{\mathcal{RD}}(x))$, and $\tilde{\mathcal{A}}(\hbar_{\mathcal{L}}(x))$.

Remark 2 is a critical observation. It indicates that for a constraint filtering algorithm to be effective, not only active line flow and ramp up/down constraints must be identified but also several other inactive constraints may need to be included in the D-ED problem. If only active constraints are used to formulate a truncated optimization problem, the resultant optimal generation schedule may differ from the original optimal schedule. Although the objective function



(a)



(b)

Figure 3.2. Block diagram of proposed classifier a) training phase and b) utilization with an embedded loop.

value does not change as generation exchange occurs among generators with the same cost coefficient, the new generation setting may activate several pseudo-active constraints and thus yields infeasibility from the original D-ED's perspective even if there are a few pseudo-active constraints. Active constraints are sufficient to formulate a truncated problem if no generators have identical cost coefficients.

Quasi-active constraints, which appear due to integer variables, are introduced in [21]. Although the concept of quasi-active constraints (i.e., not active in the optimal point but if removed changed the optimal point) is similar to what we call pseudo-active constraints, pseudo-active constraints appear mainly because of non-uniqueness of thermal unit generation cost functions. Thus, we have used the term pseudo-active to avoid confusion with the term quasi-active constraints in [21].

3.3.2. Illustrative Examples

***Example 1*:** Consider a simple economic dispatch problem (3.6) with three generators and a 2-hour horizon. The objective is to minimize generation costs with respect to power balance, generator limits, and ramp up/down constraints (branch flow constraints are ignored for simplicity).

$$\min_{p_{u,t}} 10p_{1,1} + 10p_{2,1} + 15p_{3,1} + 10p_{1,2} + 10p_{2,2} + 15p_{3,2} \tag{3.6a}$$

s.t.

$$0 \le p_{1,1}, p_{1,2}, p_{2,1}, p_{2,2}, p_{3,1}, p_{3,2} \le 200 \tag{3.6b}$$

$$p_{1,1} + p_{2,1} + p_{3,1} = 100 \tag{3.6c}$$

$$p_{1,2} + p_{2,2} + p_{3,2} = 200 \tag{3.6d}$$

$$p_{1,2} - p_{1,1} \le 50 \tag{3.6e}$$

$$p_{1,1} - p_{1,2} \le 50 \tag{3.6f}$$

$$p_{2,2} - p_{2,1} \le 60 \tag{3.6g}$$

$$p_{2,1} - p_{2,2} \le 60 \tag{3.6h}$$

$$p_{3,2} - p_{3,1} \leq 100 \qquad (3.6i)$$

$$p_{3,1} - p_{3,2} \leq 100 \qquad (3.6j)$$

This problem has multiple solutions as two generators have the same cost. An optimal solution is $p_{1,1} = 100$, $p_{2,1} = 0$, $p_{3,1} = 0$ for hour 1 and $p_{1,2} = 150$, $p_{2,2} = 50$, $p_{3,2} = 0$ for hour 2 with an objective value of \$3,000. The only active inequality constraint is $(3.6e)$. If we remove all inactive inequality constraints $(3.6f) - (3.)$, the solver may provide $p_{1,1} = 0$, $p_{2,1} = 100$, $p_{3,1} = 0$, $p_{1,2} = 0$, $p_{2,2} = 200$, and $p_{3,2} = 0$ with the objective value of \$3,000 as a solution of truncated problem $(3.6a) - (3.6e)$. This solution is not feasible from the original problem perspective as constraint $(6g)$, called pseudo-active constraint, is violated. This mathematical example shows the necessity of pseudo-active constraints to recover the optimal solution if generator cost functions are not unique.

***Example 2*:** A system with ten interconnected areas representing the IEEE 118-bus system is used to illustrate remark 2. The original D-ED is solved with all branch and ramp constraints. The number of active line flow, ramp up, and ramp down constraints are 279, 365, and 330, respectively. A truncated D-ED is formed using these active constraints and dropping other inactive constraints. The truncated D-ED is solved, and the number of newly activated line and ramp up/down constraints is 63, 155, and 134. Three hundred fifty-two newly activated (pseudo-active) constraints should also be included in D-ED. We carry out this iterative procedure. The iterative loop converges after 33 iterations. Fig. 3.3 shows the number of accumulated constraints (active plus pseudo-active) over iterations. The number of newly activated constraints usually reduces as more iterations are carried out. Also, multiple pseudo-active constraints are identified at each iteration. The total number of active (necessary) and pseudo-active (sufficient) line flow,

ramp up, and ramp down constraints required to enclose the feasible set of D-ED is 518, 1024, and 941 obtained upon the loop convergence.

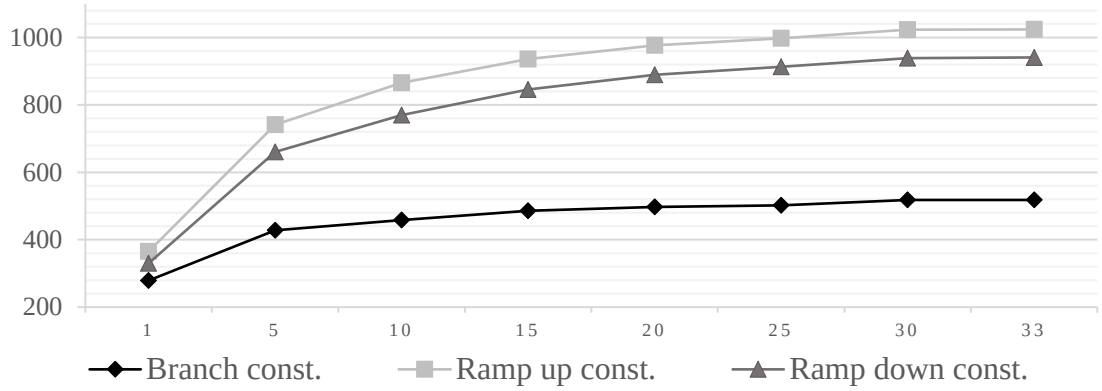


Figure 3.3. Number of accumulated active constraints over iterations.

Example 3: Consider the 118-bus system with the baseload and base topology. Generators 4, 5, 10, 29, 36, 43, 44, and 45 have the same production cost coefficient. The D-ED subproblem is formed using only active constraints, with no iterative loop. Table 3.1 shows power generation values. While the summation of power produced by these units and operational costs are the same for the original and truncated D-ED models, their generation schedules are different. This generation difference results in the violation of some originally inactive constraints that are not included in the truncated D-ED and makes its solution infeasible. For instance, the ramp-up constraint of unit 5 for the transition from interval 1 to interval 2, which is inactive in the original SCED, is activated in the truncated D-ED.

Table 3.1. Power generation values in MW

Gen. no.	4	5	10	29	36	40	43	44	45
Original D-ED	150	300	300	256	150	50	100	100	100
Truncated D-ED	226	100	100	80	300	20	100	300	100

3.3.3. Proposed Training Dataset Generation Algorithm

We present the iterative algorithm shown in the following pseudocode to capture necessary (active constraints) and sufficient (pseudo-active constraints) information of $\tilde{\mathcal{A}}(\hbar_{\mathcal{RU}}(x))$, $\tilde{\mathcal{A}}(\hbar_{\mathcal{RD}}(x))$, and $\tilde{\mathcal{A}}(\hbar_{\mathcal{L}}(x))$ for forming the truncated D-ED. Our experimental results on many cases show that when merely active constraints from the original D-ED solution are used to form the truncated problem, multiple iterations are required to find pseudo-active constraints. Our goal is to minimize the number of iterations using machine learning. We use information in the last iteration of Algorithm I to train constraint classifiers.

Algorithm I Pseudocode to capture necessary and sufficient information

For each operating condition scenario, set iteration index $k = 1$ and $flag^k = \emptyset$
Set $\mathcal{A}^k\big(h_{\mathcal{L}}(x)\big) = \mathcal{A}^k\big(h_{\mathcal{RU}}(x)\big) = \mathcal{A}^k\big(h_{\mathcal{RD}}(x)\big) = \emptyset$
Solve D-ED problem (1)
Identify active constraints $\mathcal{A}^k\big(\hbar_{\mathcal{L}}(x)\big)$, $\mathcal{A}^k\big(\hbar_{\mathcal{RU}}(x)\big)$ and $\mathcal{A}^k(\hbar_{\mathcal{RD}}(x))$
if $\mathcal{A}^k\big(\hbar_{\mathcal{L}}(x)\big) = \mathcal{A}^k\big(\hbar_{\mathcal{RU}}(x)\big) = \mathcal{A}^k\big(\hbar_{\mathcal{RD}}(x)\big) = \emptyset$
    $flag^k \leftarrow \emptyset$
else
    $flag^k \leftarrow 1$
end if
Drop (1d) – (1f) from (1) and form a D-ED subproblem
while $flag^k \neq \emptyset$
    $\tilde{\mathcal{A}}^k(\hbar_{\mathcal{L}}(x)) = \bigcup_1^k \mathcal{A}^k\big(\hbar_{\mathcal{L}}(x)\big)$
    $\tilde{\mathcal{A}}^k\big(\hbar_{\mathcal{RU}}(x)\big) = \bigcup_1^k \mathcal{A}^k\big(\hbar_{\mathcal{RU}}(x)\big)$
    $\tilde{\mathcal{A}}^k\big(\hbar_{\mathcal{RD}}(x)\big) = \bigcup_1^k \mathcal{A}^k\big(\hbar_{\mathcal{RD}}(x)\big)$
    Add $\tilde{\mathcal{A}}^k(\hbar_{\mathcal{L}}(x))$, $\tilde{\mathcal{A}}^k\big(\hbar_{\mathcal{RU}}(x)\big)$, and $\tilde{\mathcal{A}}^k\big(\hbar_{\mathcal{RD}}(x)\big)$ to the D-ED subproblem and solve it
    $k = k + 1$
    Identify pseudo-active constraints $\mathcal{A}^k\big(\hbar_{\mathcal{L}}(x)\big)$, $\mathcal{A}^k\big(\hbar_{\mathcal{RU}}(x)\big)$ and $\mathcal{A}^k(\hbar_{\mathcal{RD}}(x))$
    If $\mathcal{A}^k\big(h_{\mathcal{L}}(x)\big) = \mathcal{A}^k\big(h_{\mathcal{RU}}(x)\big) = \mathcal{A}^k\big(h_{\mathcal{RD}}(x)\big) = \emptyset$
        $flag^k \leftarrow \emptyset$
    else
        $flag^k \leftarrow 1$
    end if
end while
Store $\tilde{\mathcal{A}}^k(\hbar_{\mathcal{L}}(x))$, $\tilde{\mathcal{A}}^k\big(\hbar_{\mathcal{RU}}(x)\big)$, and $\tilde{\mathcal{A}}^k\big(\hbar_{\mathcal{RD}}(x)\big)$

Algorithm I is carried out for every operating condition scenario and accumulated active plus pseudo-active constraints obtained from iteration one to the last iteration are labeled. For a given network topology, demand and generation cost scenarios are generated using the following equations.

$$d_{nt}^{m} = \omega \times \left[d_{base,n}\left(1 - \Delta_{d}^{\mathrm{L}}\right) + \eta_{p,n} \times \left(\Delta_{d}^{\mathrm{U}} - \Delta_{d}^{\mathrm{L}}\right)\right] \qquad (3.7)$$

$$\gamma_{nt} = \left[\gamma_{base,n}\left(1 - \Delta_{\gamma}^{\mathrm{L}}\right) + \eta_{a,n} \times \left(\Delta_{\gamma}^{\mathrm{U}} - \Delta_{\gamma}^{\mathrm{L}}\right)\right] \qquad (3.8)$$

Parameters Δ_{d}^{L} and Δ_{d}^{U} are the upper and lower bounds of demand variation for each load curve, and Δ_{a}^{U} and Δ_{a}^{L} are upper and lower bounds of generation cost variation. Random parameter $\eta_{a,n}$ follows a uniform distribution between 0 and 1. It adds randomness to the base case generation cost γ_{base} at bus n. Two randomness factors are introduced for demand to capture various plausible operating conditions. Random parameter ω shifts the load curve to model its daily and seasonal variation. It can be varied in a range using historical data and load growth predictions or such that any further increment/decrement makes economic dispatch infeasible. Random parameter $\eta_{p,n}$, which follows a uniform distribution between 0 and 1, models the uncertain geographic load distribution. Parameter $\eta_{p,n}$ is generated for every load point, allowing load at different buses to fluctuate independently. Consider the base case load curve on the left side of Fig. 3.4. It would become similar to right side curves after adding load cure shifting randomness ω and nodal load distribution randomness $\eta_{p,n}$.

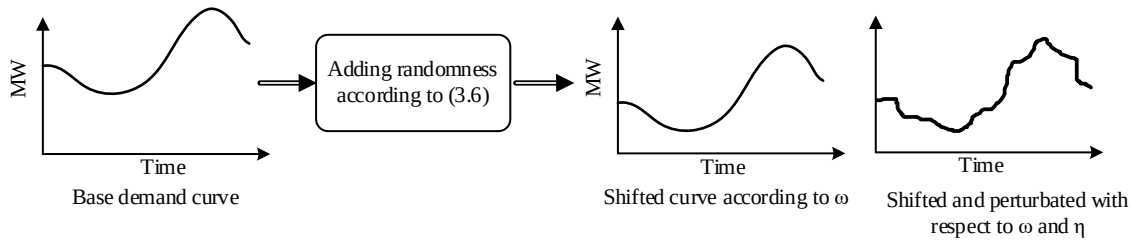


Figure 3.4. Demand scenario generation.

A major bottleneck of existing learning-based constraint classification approaches is that learners are trained for a fixed topology. But in real-word, the network topology changes frequently. A new learner would be required every time the network topology alters. We use the admittance matrix information to address this problem. A topology alteration changes some buses' self-impedance which can be detected by observing diagonal elements of the admittance matrix. We use (7) and (8) to generate a set of demand and generation cost scenarios for every topology configuration. This leads to the following operating condition matrix ($\mathcal{OC}_t$) at time period t. The first and second columns of $\mathcal{OC}_t$ contain nodal demand and generation costs, and its third column is the admittance matrix diagonal elements.

$$\mathcal{OC}_t = \begin{bmatrix} d_{1t} & \gamma_{1t} & Y_{11,2} \\ d_{2t} & \gamma_{2t} & Y_{22,t} \\ \vdots & \vdots & \vdots \\ d_{nt} & \gamma_{nt} & Y_{nn,t} \end{bmatrix} \tag{3.9}$$

where n is the number of buses. An operating condition scenario is formed by combining $\mathcal{OC}_t$ for all scheduling periods $t = 1, \dots, T$. The D-ED problem (1) is solved for each operating condition scenario. Inequalities (3.1d)–(3.1f) are dropped from (1), and a D-ED subproblem is formed using active generator ramping limitations and line flow constraints (3.3)–(3.5). This D-ED subproblem is solved. If new active ramp up/down or line flow constraints are observed, $flag^k \leftarrow 1$, and an iterative loop is started. At each iteration k, all active and pseudo-active constraints from iteration

1 to $k-1$ are added to the D-ED subproblem. The loop is carried out, and active and pseudo-active constraints are accumulated. If no newly activated constraint is detected and $flag^k = \emptyset$, the accumulated constraints $\tilde{\mathcal{A}}^k(\hbar_{\mathcal{L}}(x))$, $\tilde{\mathcal{A}}^k\left(\hbar_{\mathcal{RU}}(x)\right)$, and $\tilde{\mathcal{A}}^k\left(\hbar_{\mathcal{RD}}(x)\right)$ are stored for training classification learners.

3.3.4. Operating Condition Conversion into DCT Colorful Image

Knowing that i) a node/line in a power system interacts with its neighboring nodes and transmission lines and is loosely coupled with distant nodes and lines [47] and ii) a pixel of an image is highly correlated to its neighboring pixels, we obtain the intuition to convert the constraint screening classification into a computer vision type problem. We use a 3-D tensor to convert a power system operating condition scenario into a colorful image. The matrices of this tensor corresponding to red, green, and blue color channels contain, respectively, demand, thermal units' generation cost, and network topology information. We call this image a DCT image (D: demand, C: cost, and T: topology).

Demand and generation cost terms are extended to every bus to have the same-sized matrices. The demand/generation cost input for every time period is a vector with n elements that are set to zero for buses with no generators or no load. Hence, demand matrix $\mathcal{D}$ and cost coefficient matrix Γ are $n \times T$, where T is the considered scheduling horizon.

$$\mathcal{D} = \begin{bmatrix} d_{11} & \cdots & d_{1T} \\ \vdots & \ddots & \vdots \\ d_{n1} & \cdots & d_{nT} \end{bmatrix} \tag{3.10}$$

$$\Gamma = \begin{bmatrix} \gamma_{11} & \cdots & \gamma_{1T} \\ \vdots & \ddots & \vdots \\ \gamma_{n1} & \cdots & \gamma_{nT} \end{bmatrix} \tag{3.11}$$

We use diagonal elements of the admittance matrix at every time period and form the following $n \times T$ matrix.

$$\mathcal{Y} = \begin{bmatrix} Y_{11,1} & \cdots & Y_{11,T} \\ \vdots & \ddots & \vdots \\ Y_{nn,1} & \cdots & Y_{nn,T} \end{bmatrix} \tag{3.12}$$

Consider an operating condition for the IEEE 118-bus system with a scheduling horizon of 24 periods. The red channel of the DCT image, i.e., $\mathcal{D}$, is a 118×24 matrix with each column having 91 nonzero elements and 27 zeros. The green channel matrix Γ has 54 nonzero elements and 64 zeros in each column. As all diagonal elements of the Y-bus matrix are nonzero, all elements of blue channel $\mathcal{Y}$, whose size is 118×24, are nonzero. A DCT image for an operating condition scenario is shown in Fig. 3.5.

The locations of zeros added to $\mathcal{D}$ and Γ matrices are fixed. This zero padding does not affect the training time and learners' performance. As explained in the next section, the output after convolution and pooling operations of the zero-padded pixels is zero. As a result, neuron weights corresponding to zero pixels are not updated after backpropagation.

Any alteration in one or a combination of demand, cost, and topology-related tensors changes the DCT image. A well-trained convolutional neural network (CNN) can capture even slight pattern changes in an image. Furthermore, system features and active constraints do not vary drastically after, for instance, a topology alteration. Thus, features learned by CNN before and after the outage of a line l can help the learner predict the status of the constraints if a line near line l is out.

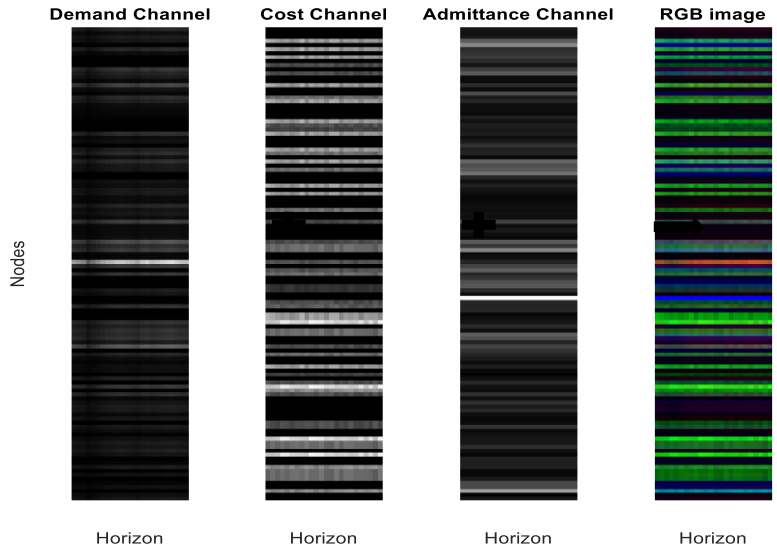


Figure 3.5. Red, green, and blue channels of DCT image corresponding to an operating condition scenario for the 118-bus system.

3.3.5. Learning Strategy

CNN has shown promising performance in image analysis and computer vision problems. CNN is consistently performing better and has become the state-of-the-art object detection and image classification technique. We have tailored the considered constraint classification problem as a computer vision problem and have selected CNN to tackle it.

3.3.5.1. CNN Classifier

CNN extracts meaningful local features through repeated convolution/pooling operations. CNN exploits shift-invariance, local connectivity, and compositionality. By carefully organizing the input shape of CNN, it is possible to exploit the power system temporal and geographical dependency information. The weights of learnable neurons are updated by interacting with three dimensions (i.e., demand, cost, and admittance) of a DCT image. Every neuron in a layer is correlated to a small region of the preceding layer instead of all neurons. Batches of images of a particular shape are fed to CNN to extract feature vectors through convolution. After every

convolution, the number of extracted features depends on the number of filters. The trainable layer parameters are optimized using a loss function.

3.3.5.2. Learner Architecture

Instead of training a CNN learner from scratch, we have selected EfficientNet-B7, a CNN-based pre-trained model developed by the Google Brain Team. EfficientNet-B7 is one of the latest state-of-the-art developments in the image classification domain [48]. EfficientNet-B7 attains 84.3% top-1 and 97.1% top-5 accuracy with 66M parameters and 37B FLOPS (floating point operations per second), whereas the earlier best GPipe achieves similar performance with 557M parameters while being 8.4 times larger than EfficientNet-B7 [48].

Hyperparameters play a significant role in CNN efficiency and accuracy. Effective scaling/hyperparameter tuning is still an open question [48]. EfficientNet-B7 tuning follows a compound scaling method. It provides a compound coefficient to uniformly scale network width (number of channels), depth (number of layers), and image resolution together instead of independently scaling each parameter. EffcientNet-B7 developers have already set these parameters through extensive experimentation. This pre-trained model reduces the need for setting many hyperparameters. Also, features learned by this pre-trained model can help enhance the accuracy and efficiency of the branch and ramp constraint classification problem. This architecture can serve as a foundation for power system optimization problems that can be converted into computer vision problems.

3.3.5.3. Transfer Learning

Transfer learning refers to utilizing features learned from a problem and leveraging them for a new problem to improve learning performance and accuracy. We propose exploiting pre-trained

EfficientNet-B7 and using transfer learning to adapt it with the considered constraint classification problem. Such pre-trained models contain important features preserved in a feature space and transferable to other tasks. Three alternatives exist: 1) reusing the trained weights of one or more layers of a pre-trained network. 2) Fine-tuning all layers entirely for a new dataset (a weight initialization scheme using pre-trained weights). 3) Keeping pre-trained weights fixed and adding new layers on top of the pre-trained network.

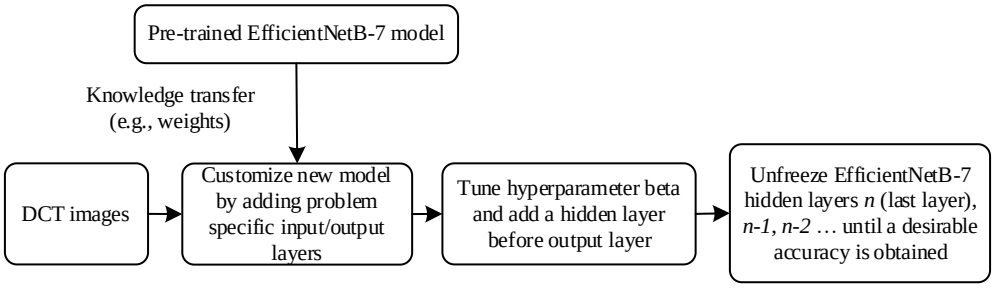


Figure 3.6. Block diagram of the proposed transfer learning procedure.

Fig. 3.6. shows the concept of transfer learning that follows several steps, as shown in Algorithm II. We remove the output layer of EfficientNet-B7 and add a customized output layer whose size depends on the number of branch and ramp constraints. We also add a hidden layer before the output layer. The added hidden and output layers will transform the old features into predictions on a new dataset. We fine-tune the weights of the last few pre-trained layers (i.e., layers before the added new layers) of EfficientNet-B7, as these final layers capture more data-specific features. One can unfreeze some (e.g., three) last hidden layers before the output layer or unfreeze the last hidden layer and increase the number of unfrozen last hidden layers until a desirable accuracy is obtained. These layers are fine-tuned at a low learning rate with the new DCT images representing power system operating conditions.

Algorithm II Pseudocode to transfer learning

1. Import weights of a pre-trained EfficientNet-B7 model
2. Remove the top output layer
3. Freeze layers to avoid destroying learned features
4. Add a new trainable output layer on top of frozen layers
5. Add a new trainable hidden layer before the output layer
6. Train only new layers on using DCT image datasets representing power system operating conditions (a few epochs)
7. Unfreeze the last hidden layer and train the model at a very low learning rate (several epochs)
8. Repeat Step 7 by unfreezing the last hidden layers one by one until desirable accuracy is obtained

3.3.5.4. Loss Function

The considered constraint screening problem is a binary classification. A common loss function for binary classification problems is binary cross-entropy. However, this function may not be suitable for the line and ramp constraints classification problem as the dataset is unbalanced. The numbers of active and inactive constraints are not in the same order. Most constraints are inactive for the power system to comply with North American Electric Reliability Corporation (NERC) standards. On the other hand, the conventional accuracy metric is interpretable but not robust against uneven data and can yield misleading evaluation.

We have used the $F\beta score$ loss function with a customized β to reduce the impact of unbalanced data. A loss function should be continuous and differentiable for learning optimization problems. $F\beta score$, which is a discrete value, is modified to make it differentiable.

$$F\beta score = (1+\beta^2)\frac{Precision * Recall}{\beta^2 * Precision + Recall}$$

$$= \frac{(1+\beta^2) * TP}{(1+\beta^2) * TP + FP + \beta^2 * FN)} \quad (3.13)$$

where precision, recall, TP, TN, FP, and FN metrics are:

- ***True positives*** (TP): Actual and predicted status is ACTIVE
- ***True negatives*** **(TN):** Actual and predicted status is INACTIVE.
- ***False positives*** (FP): Actual status is INACTIVE, and predicted status is ACTIVE (type I error).
- ***False negatives*** **(FN):** Actual status is ACTIVE and predicted status is INACTIVE (type II error).

$$Precision = \frac{TP}{FP + TP} \quad (3.14)$$

$$Recall = \frac{TP}{TP + FN} \quad (3.15)$$

3.3.5.5. Hyperparameter Tuning

Many important hyperparameters are set by EffcientNet-B7 developers. We only need to set β, the number of neurons in the newly added hidden layer before the output layer, and the number of EffcientNet's hidden layers that should be unfrozen. Hyperparameter β in (3.13) controls the importance of precision and recall and is usually tuned through experiment. $\beta < 1$ (e.g., 0.5) assigns more weight to precision and less weight to recall. $\beta = 1$ assigns the same weight to both precision and recall. $\beta > 1$ (e.g., 2) gives less weight to precision and more weight to recall. This approach is suitable when both precision and recall carry similar significance, but more attention is needed on false negatives. The number of neurons in the newly added hidden layer can be set as a 2^n number closets to the number of neurons in the output layer.

3.3.6. Line flow, Ramp Up, and Ramp Down Classifiers

We train three classifiers, one for each constraint type, instead of training a single classifier for all constraints. The first classifier is dedicated to line flow constraint status identification. The second classifier is devoted to generating unit ramp up limitations, and the third classifier is dedicated to ramp down constraints. This strategy can enhance the constraint classification accuracy and speed up the training process through parallel training.

3.4. Numerical simulation and results analysis

The proposed algorithm is tested on the EEE 24-bus system, the IEEE 118-bus system, and the 6515-bus French system. The considered scheduling horizon has 24 time periods. The YALMIP toolbox and IBM-ILOG-CPLEX are used to model and solve D-ED [40, 49]. The Python-based Keras framework is used with TensorFlow for machine learning. Simulations are carried out on a computer with Intel(R) Xeon(R) 2.10 GHz CPU and 512 GB of RAM. We have posted our code on GitHub [49].

3.4.1. Active and Pseudo-active Constraint Statistics

Many operating condition scenarios are generated for each test system. The average percentage of active and pseudo-active constraints required to form the D-ED feasible region is reported in Table 3.2. For instance, for the 118-bus system, the original D-ED problem has $186 \times 24 = 4463$ branch constraints, $54 \times 23 = 1242$ ramp up constraints, and 1242 ramp down constraints. On average, the number of active branch, ramp up, and ramp down constraints are 4, 47, and 45, respectively. Sets $\tilde{\mathcal{A}}^k(\hbar_{\mathcal{L}}(x))$, $\tilde{\mathcal{A}}^k\left(\hbar_{\mathcal{RU}}(x)\right)$, and $\tilde{\mathcal{A}}^k\left(\hbar_{\mathcal{RD}}(x)\right)$ include 10, 76, and 71 active and pseudo-active constraints. Pseudo-active constraints form a large percentage of $\tilde{\mathcal{A}}^k(\hbar_{\mathcal{L}}(x))$, $\tilde{\mathcal{A}}^k\left(\hbar_{\mathcal{RU}}(x)\right)$, and $\tilde{\mathcal{A}}^k\left(\hbar_{\mathcal{RD}}(x)\right)$, without which the feasible design region of the truncated D-ED

is not the same as that of the original D-ED. Table 3.2 shows that a larger percentage of generating unit ramp constraints is required than branch constraints to formulate the truncated D-ED. A similar trend is observed for the 6515-bus French system, whose 19 generating units do not have a unique generation cost function.

Table 3.2. Average percentage of active and pseudo-active constraints

System	Active constraints			Active + pseudo-active constraints		
	Branch	RU	RD	Branch	RU	RD
Case24	0.47	3.5	2.9	0.53	3.6	3.0
Case118	0.12	3.9	3.75	0.26	6.4	5.9
Case 6515	0.3	18.3	9.7	0.06	20.7	10.7

3.4.2. Dataset Preparation and Learners Architecture

Algorithm I is implemented to generate training datasets. Nodal demand and cost are varied within a range to generate scenarios for each plausible network topology. Table 3.3 shows the perturbation range for each system as compared to the base case values. D-ED is solved for each scenario. Active and pseudo-active constraints are labeled as 1, and the rest are labeled as 0. The operating condition scenarios are converted into the DCT image format. We assign a branch label set, a ramping up label set, and a ramping down label set for each DCT image. Three classifiers are trained whose input is DCT images. The target of classifiers 1, 2, and 3 are the branch label set, the ramping up label set, and the ramping down label set. Classifier parameters are given in Table 3.4. To form each classifier, the EfficientNet-B7 architecture is imported along with its weights, excluding the top layer. This truncated architecture becomes the base model. A new model is created using the transfer learning concept by adding a customized output layer to the base model to comply with the new learning tasks (e.g., learning branch classification).

Historical data should be collected to utilize the proposed algorithm for large real-world systems. Redundant unique samples should be dropped. If the dataset is large, various scenario clustering and reduction techniques (e.g., K-means) can be implemented. Similar scenarios can be grouped in the same cluster, and one or multiple representatives from each class can be selected. Although the training might take time for large systems, it is an offline procedure carried out once.

Table 3.3. Variation range

System	Load		Cost coefficient	No. of scenario	
	ω	Δ_d^{L} to Δ_d^{U}	$\Delta_b^{\mathrm{U}}, \Delta_b^{\mathrm{L}}$	Train	Test
Case24	70% -130%	97%-103%	$\pm$ 15%	4000	1000
Case118	90%-119%	97%-103%	$\pm$ 15%	4000	1000
Case6515	80%-119%	97%-103%	$\pm$ 15%	3010	700

Table 3.4. Hyperparameter of modified EfficientNet-B7 architecture: Added layers after flattening

	Final FC layers and Training parameters	
Classifiers (Branch, RU, RD)	Added hidden layer=1, Batch size=500, Validation split=10%, Early stopping with Patience 10 (min. no. of epochs)	Activation =ReLU & Sigmoid Loss & metric= F2 Optimizer= Adam

3.4.3. Prediction Analysis

The size of the test datasets is given in Table 3.3. The constraint statuses predicted by classifiers are compared with ground truth data obtained by solving D-ED. As the power system is safety-critical, we analyze false negatives and false positives performance statistics. The classification accuracy depends on hyperparameter values, such as β and the number of training epochs. The percentage of FPs and FNs can be controlled by tuning hyperparameters. Since having constraints that are active but classified as inactive is undesirable, we set $\beta = 2$, meaning that recall is twice as important as precision. This reduces the number of FNs. Table 3.5 shows the average FP and

FN percentages. In general, more ramping constraints are misclassified in the FP category than line constraints. This might be because intertemporal connectivity makes classifying ramp constraints more complex than line flow constraints. Some unnecessary constraints are added to the truncated D-ED due to FPs. This increases the size of the truncated D-ED. Having fewer FNs is more crucial as they include the necessary information to form a feasible design space. A few FNs are observed that will be added to the optimization constraints using the iterative loop. The prediction error is inevitable, and thus removing this loop would make the truncated economic dispatch solution infeasible for test scenarios with nonzero FNs.

Table 3.5. Average FPs and FNs per scenario

System	False negative (FN)			False positive (FP)		
	branch	RU	RD	branch	RU	RD
Case24 (Eff)	0.31	7.0	2.9	17.5	4.4	40
Case24 (NN)	1.1	6.1	6.3	1.4	10.3	9.0
Case24 (CNN)	0.2	6.4	6.6	22.7	20.0	3.2
Case118 (Eff)	1.3	2.3	5.6	151	370	278
Case118 (NN)	0	0	0	12.4	78.3	72.3
Case118 (CNN)	1.7	60	54	278	841	405
Case6515 (Eff)	2.8	8.4	4.5	133	4156	4399
Case6515 (NN)	0.2	0.03	0.03	256	7636	7524

One can select a smaller β to reduce FPs. But it would increase FNs, and thus the number of iterations and overall solution time. In the worst-case scenario, the number of iterations would be equal to the number of branch and ramp constraints minus FNs. However, it would not happen as not all branch and ramp constraints are active. Also, our observations (Table 3.6) show that several pseudo-active constraints are added to optimization after carrying out each iteration.

We have also trained a CNN and a neural network (NN). Generally speaking, these two learners have more FP misclassifications. Although we have used the pre-trained EffcientNet-B7, one can use NN and CNN. One of the advantages of EffcientNet-B7 is its pre-trained know structure. A

user does not need to make significant changes in the learner structure, such as the number of hidden layers and neurons. Unlike CNN and NN, for which the best learner structure should be found based on many trials, a user needs only to change the size of output and input layers of the pre-trained EffcientNet-B7 based on the considered power system size.

3.4.4. Truncated D-ED Runtime Analysis and Solution Quality

We use an integrality gap index to show how close are the solutions of the truncated and original D-EDs [38]. f^{T-DED} and f^{D-ED} are, respectively, objective values obtained from the proposed algorithm and the original D-ED problem.

$$\text{Integrality Gap\%} = \frac{|f^{T-DED} - f^{D-ED}|}{f^{D-ED}} \times 100 \qquad (3.16)$$

The average integrality gap for all test scenarios is negligible (less than 10^{-7}) for all three test systems, showing that the proposed truncated D-ED algorithm provides the same solution as the original D-ED.

The computation time saving obtained by the proposed D-ED algorithm is reported in Table 3.6. The average number of iterations and time over test scenarios are reported. For the 118-bus system, for instance, the iterative loop converges after 2.07 iterations on average. This is due to the FN misclassifications. Without the iterative loop, the truncated D-ED solution may become infeasible as a few necessary active or pseudo-active constraints are missed in the truncated constraint set. The time saving becomes promising as the system size increases. While no time saving is observed for the 24-bus system, the optimization solution time is reduced by 99% for the 6515-bus system.

Table 3.6. Performance analysis: Average iteration numbers and time-saving

Systems	Original D-ED time	Truncated D-ED		Time saving
		No. iter	Total time	
Case24	11 ms	1.82	14 ms	No save
Case118	220 ms	2.07	140 ms	32%
Case6515	238 sec	1.69	< 1 sec	99%

3.4.5. Comparison with ICG

We compare the proposed approach with ICG using the 6515-bus system. ICG is a popular method in which constraints are relaxed, a master problem is formulated and solved, and violated constraints are added to the master problem iteratively. Simulations are run for all test operating condition scenarios, and average values are reported in Table 3.7.

Table 3.7. Comparison with ICG

Test system	Average solver time (sec.)	Average number of iterations
Original D-ED (benchmark)	238	-
ICG	64	50.4
Proposed approach	< 1	1.69

The original D-ED problem with all constraints is solved to obtain benchmark results. It takes 238 seconds. The proposed approach takes much fewer iterations and less time than ICG to find the optimal D-ED solution. The proposed approach is 98% faster than ICG.

3.4.6. Memory Usage Analysis and Runtime Comparison for Combined Branch and Ramp Constraints Screening

The average memory requirement (MB) for building the constraint set and solving time is reported in Table 3.8 for only branch constraints screening and branch and ramp constraints screening. The least memory usage is observed after screening both branch and ramp constraints and dropping inactive constraints from the model. For instance, for the 6515-bus system, the

original D-ED problem occupies 5489 MB of memory. It reduces almost 37 times by dropping inactive branch constraints and 211 times if both inactive branch and ramp constraints are dropped. While screening only branch constraints leads to better time saving for smaller systems, screening both branch and ramp constraints saves more time for larger systems. For the 6515-bus system, branch and ramp constraints screening achieve 30% more time-saving. Since screening only branch constraints would lead to a good time saving even for large systems, one may ignore ramp constraints screening. However, we suggest branch and ramp constraints screening for larger systems to reduce memory usage significantly. For instance, for the 6515-bus system, screening both branch and ramp constraints results in better time saving and a significant RAM requirement reduction. It thus makes solving large systems possible even without the need for supercomputers with large memory.

Table 3.8. Average memory requirement (MB) to build constraints and solver time for two constraints screening schemes

System	Original problem	Screening branch constraints		Screening branch & ramp constraints	
	RAM	RAM	Time	RAM	Time
Case24	7	2.5	10 ms	0.35	14 ms
Case118	44	12.4	20 ms	1.2	140 ms
Case6515	5489	145	1 sec	26	0.7 sec

3.4.7. Demand vs. DCT as Learner Input

Under a given demand value, thermal units' generation cost coefficients and grid topology may differ, resulting in different active/inactive constraint sets. If demand is used as the only input feature, the learner may face difficulty predicting the status of constraints. More misclassifications may increase the problem size and number of loop iterations and thus the solver time. Table 3.9 shows the number of loop iterations and solver time. We suggest using DCT as the learner input

to reduce the solver time and required memory usage. However, one can use only demand since the embedded iterative loop can eventually capture all required constraints to form a truncated D-ED.

Table 3.9. Time gain analysis using demand and DCT as learner's input

Systems	Truncated D-ED (only demand)		Truncated D-ED (DCT image)	
	No. iter	Total time	No. iter	Total time
Case24	1.8	14 ms	1.82	14 ms
Case118	3.8	192 ms	2.07	140 ms
Case6515	1.71	< 1 sec	1.69	< 1 sec

3.4.8. Hamming Distance Analysis

Hamming distance measures the difference between two binary strings. It is an indicator of output features' sensitivity to input features and the robustness of the proposed algorithm. We have perturbed the demand, identified active constraints for two consecutive demand scenarios, and calculated Hamming distance between the active constraint status indicators, which are 0/1strings. Table 3.10 shows the average Hamming distance for branch and ramp constraints. The sensitivity of output features to input features is not high. Hamming distances corresponding to branch constraints are lower than those of ramp constraints. As shown in Table 3.5, this could justify observing more ramp constraint misclassifications than branch constraints.

Table 3.10. Hamming distance analysis

System	Branch	Ramp up	Ramp down
Case24	0.34%	5.19%	4.8%
Case118	0.21%	6.21%	5.64%
Case6515	0.013%	2.02%	1.69%

3.5. Conclusion

A small subset of inequality constraints contains enough information to form the dynamic economic dispatch feasible region. This paper presents a learning-aided iterative algorithm to identify active and pseudo-active branch flow and thermal unit ramp up/down constraints required to form the D-ED feasible space for each operating condition scenario. Three classifiers are trained, one for each type of constraint, taking into consideration network topology. Using these classifiers' predictions, a truncated D-ED is formed that is smaller and less computationally expensive than the original D-ED problem.

The number of iterations of the learning-aided approach is much less than the classical ICG. Also, filtering active and pseudo-active constraints reduces iterations much more than filtering only active constraints. The benefit of constraint filtering is more significant for larger systems. The average runtime saving for the 6515-bus system is 99%. We have observed that filtering both branch and ramp constraints would lead to better time-saving and memory usage than filtering only branch constraints. However, the learning-aided approach can filter out only branch constraints. This would result in a good enough time and memory usage saving for large systems.

Chapter 4
Accelerating L-Shaped Two-Stage Stochastic SCUC With Learning Integrated Benders Decomposition

4.1. Introduction

Benders decomposition is widely used to solve large mixed-integer problems. This work takes advantage of machine learning and proposes enhanced variants of Benders decomposition for solving two-stage stochastic security-constrained unit commitment (SCUC). The problem is decomposed into a master problem and subproblems corresponding to a load scenario. The goal is to reduce the computational costs and memory usage of Benders decomposition by creating tighter cuts and reducing the size of the master problem. Three approaches are proposed, namely regression Benders, classification Benders, and regression-classification Benders. A regressor reads load profile scenarios and predicts subproblem objective function proxy variables to form tighter cuts for the master problem. A criterion is defined to measure the level of usefulness of cuts with respect to their contribution to lower bound improvement. Useful cuts that contain the necessary information to form the feasible region are identified with and without a classification learner. Useful cuts are iteratively added to the master problem, while non-useful cuts are discarded to reduce the computational burden of each Benders iteration. Simulation studies on multiple test systems show the effectiveness of the proposed learning-aided Benders decomposition for solving two-stage SCUC as compared to conventional multi-cut Benders decomposition. Additionally, our strategy incorporates an inactive constraints removal technique in the sub-problem to accelerate it and generate a training dataset more efficiently. Fig. 4.1 presents the concept of useful cut filtering.

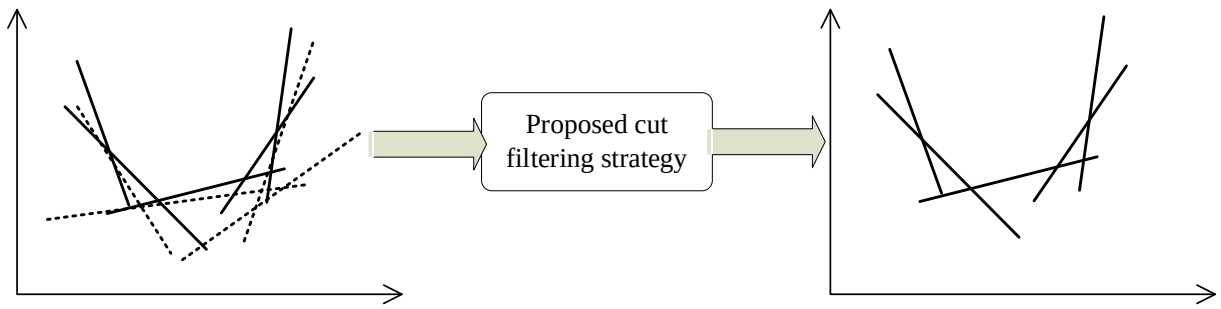


Figure 4.1. Useful cut identification for benders decomposition

4.2.Two-Stage Stochastic Unit Commitment

The considered problem is a two-stage stochastic unit commitment with respect to demand uncertainty. Benders decomposition is used to solve the problem.

4.2.1. Problem Formulation

The problem is formulated in (1), which includes two sets of variables: here-and-now and wait-and-see decisions [50]. On/off status of units are here-and-now variables, and generation dispatches are wait-and-see variables. The first term of (4.1a) is the first-stage startup and shutdown costs, and the second term is the second-stage generation dispatch costs. The first-stage constraints are (4.1b) – (4.1i), which are scenario-independent unit commitment constraints. The second-stage scenario-dependent operational constraints are formulated in (4.1j)–(4.1n). $N-1$ security constraints are included in the model where g, t, ω, c are indices for generator units, time horizons, stochastic scenarios, and contingency respectively.

$$\min \sum_{t}\sum_{g}\left(SU_g y_{gt} + SD_g z_{gt}\right) + \sum_{\omega}\pi_{\omega}\sum_{t}\sum_{g} f\left(p_{gt\omega}\right) \qquad (4.1a)$$

s.t.

$$y_{gt} - z_{gt} = u_{gt} - u_{g(t-1)} \qquad \forall g,\ \forall t \qquad (4.1b)$$

$$y_{gt} + z_{gt} \leq 1 \qquad \forall g, \forall t \qquad (4.1c)$$

$$\sum_{t=1}^{UT_g} \left(1 - u_{gt}\right) = 0 \qquad \forall g \qquad (4.1d)$$

$$\sum_{\tau=t}^{t+T_g^{on}-1} u_{g\tau} \geq T_g^{on} y_{gt} \qquad \forall g, t = UT_g + 1, UT_g + 2, \dots, T - T_g^{on} + 1 \quad (4.1e)$$

$$\sum_{\tau=t}^{T} (u_{g\tau} - y_{gt}) \geq 0 \quad \forall g, t = T - T_g^{on} + 2, \dots, T \qquad (4.1f)$$

$$\sum_{t=1}^{DT_g} u_{gt} = 0 \qquad \forall g \qquad (4.1g)$$

$$\sum_{\tau=t}^{t+T_g^{off}-1} (1 - u_{g\tau}) \geq T_g^{off} z_{gt} \quad \forall g, t = DT_g + 1, DT_g + 2, \dots, T - T_g^{off} + 1 \quad (1h)$$

$$\sum_{\tau=t}^{T} (1 - u_{g\tau} - z_{gt}) \geq 0 \quad \forall g, t = T - T_g^{off} + 2, \dots, T \qquad (4.1i)$$

{

$$P_g^{min} u_{gt} \leq p_{gt\omega c} \leq P^{max}{}_g u_{gt} \qquad \forall g, \forall t \qquad (4.1j)$$

$$p_{gt\omega c} - p_{g(t-1)\omega c} \leq RU_g\left(1 - y_{gt}\right) + P_g^{min} y_{gt} \quad \forall g, \forall t \quad (1k)$$

$$p_{g(t-1)\omega c} - p_{gt\omega c} \leq RD_g\left(1 - z_{gt}\right) + P_g^{min} z_{gt} \quad \forall g, \forall t \quad (1l)$$

$$\sum_{g} p_{gt\omega c} = D_t \qquad \forall t \qquad (4.1m)$$

$$-PL^{max} \leq SF_c \times P_{t\omega c}^{inj} \leq PL^{max} \qquad \forall t \qquad (4.1n)$$

$\} \ \forall \omega \in \Omega, \ \forall c$

4.2.2. Benders Decomposition

The computational complexity of mixed-integer program (1) increases with the increasing size of the network, number of demand scenarios, and number of contingencies. Benders decomposition is a suitable approach for solving such MIP problems with a block structure over contingencies and uncertainty scenarios[51]. We use a multi-cut variant of Benders decomposition given its faster convergence rate than the classical single-cut approach [35, 52]. Multi-cut Benders decomposition (MCBD) is an extension of single-cut Benders decomposition designed to solve mixed-integer programs. MCBD decomposes MIP problems into a set of subproblems generating multiple cuts in each iteration. This multi-cut strategy improves convergence performance [53]. Problem (1) is decomposed into a master problem and n subproblems. At each Benders iteration k, the master problem (MP) is formulated in $(2a) - (2c)$.

$$J_{MP} = \min_{x} \sum_{g} \sum_{t} \left(SU_g y_{gt} + SD_g z_{gt}\right) + \sum_{\omega} \alpha_{\omega} \qquad (4.2a)$$

s.t

$$(1b) - (1i) \qquad (4.2b)$$

$$\cup_{i=1}^{i=k-2} \hbar_{\beta}^{n}(u_{gt}, \alpha_{\omega}) \qquad (4.2c)$$

$$\hbar_{\beta}^{k-1}(u_{gt}, \alpha_{\omega}) : \alpha_{\omega} \geq J_{SP,\omega}^{k-1} + \sum_{g} \sum_{t} \lambda_{gt\omega}^{k-1} \left(u_{gt} - u_{gt}^{k-1}\right) \quad \forall \omega \qquad (4.2d)$$

$$\alpha_\omega \geq \alpha_\omega^{min} \qquad \forall \omega \qquad (4.2e)$$

$$x \in \{u_{gt}, y_{gt}, z_{gt}, \alpha_\omega\}$$

The objective function $(4.2a)$ consists of start-up and shut-down costs and a term as a proxy for subproblems' objective functions. Equation $(4.2c)$ is the accumulation of all cuts generated up to iteration $k-2$, $(4.2d)$ denotes the Benders cuts generated at iteration $k-1$, and Inequality $(4.2e)$ defines the bound on the proxy variable.

Benders subproblem ω (SP_ω) is formulated in $(4.3a)$- $(4.3e)$. To gain extra computational efficiency, we use an always-feasible subproblem model [54]. Unlike the conventional approach that generates feasibility and optimality cuts, the always-feasible model includes non-negative slack variables v and γ to prevent SP_ω infeasibility.

$$J_{SP,\omega} = \min_x \sum_t \sum_g \sum_\omega (\pi_\omega f(p_{gt\omega}) + v_{gt\omega}) + \sum_t \sum_\omega \gamma_{t\omega} \quad (4.3a)$$

$$\text{s.t} \qquad u_{gt} = u_{gt}^* \qquad (4.3b)$$

{

$$(1j) - (1n) \qquad (4.3c)$$

$$P^{min}{}_g u \leq p_{gt\omega c} + v_{gt\omega} + \gamma_{t\omega} \leq P_g^{max} u_{gt} \qquad \forall g, \forall t \qquad (4.3d)$$

$$v_{gt\omega} \geq 0, \gamma_{t\omega} \geq 0 \qquad (4.3e)$$

$$x \in \{p_{gt\omega}, v_{gt\omega}, \gamma_{t\omega}\}$$

} $\forall c$

The objective function (4.3a) consists of the generation costs and constraint violation costs associated with auxiliary variables $v_{gt\omega}$ and $\gamma_{t\omega}$. Constraint $(4.3b)$ sets the first stage decisions, i.e., generator unit status, as fixed values received from the master problem. Inequalities $(4.3e)$ set bound on auxiliary variables. The MP and SPs are solved iteratively until convergence tolerance ε is smaller than a predetermined threshold.

$$\varepsilon = \frac{UB - LB}{abs(LB)} \tag{4.4}$$

where

$$LB = J_{MP}$$

$$UB = J_{SP} + J_{MP} - \sum_{\omega} \alpha_{\omega}$$

The two-stage stochastic SCUC with multi-cut Benders is summarized in Algorithm I.

Algorithm I Two-stage stochastic SCUC with multi-cut Benders

1. Initialize convergence tolerance ε, set iteration index $k = 0$, and set α_{ω}^{min}
2. **while** ε > tolerance
3. $k = k + 1$
4. Solve master problem (4.2) to obtain u_{gt}^{k}
5. **for** $\boldsymbol{\omega = 1:n}$
6. Solve SP_{ω} (3) to obtain $J_{SP,\omega}^{k}$ and $\lambda_{gt\omega}^{k}$
7. **end for**
8. Calculate convergence tolerance $\varepsilon = \frac{UB-LB}{abs(LB)}$
9. Form new cuts as $\alpha_{\omega} \geq J_{SP,\omega}^{k} + \sum_{g}\sum_{t} \lambda_{gt\omega}^{k} \left(u_{gt} - u_{gt}^{k}\right) \forall \omega$ and go to Step 2
10. **end while**

4.3. Proposed Learning Assisted Benders Decomposition

While multi-cut Benders decomposition outperforms single-cut Benders in terms of convergence performance, its memory usage and computational overhead still need enhancement.

We present three learning-aided approaches to reduce the computational costs of multi-cut Benders decomposition for solving two-stage stochastic SCUC. We mainly focus on the master problem, which needs higher computational costs than subproblems.

4.3.1. Regression-assisted Benders Decomposition (R-Benders)

The first proposed approach uses a combination of a regressor and cut filtering. The regressor aims at forming a tighter bound for the master problem at the root node starting from $k = 1$. We predict the optimal subproblem proxy variables α_ω^*, i.e., α_ω upon the convergence of Algorithm I. Knowing α_ω^*, or even a good approximation, makes inequalities (4.2c), (4.2d), and (4.2e) tighter and the lower bound LB much closer to the optimal subproblem cost. The cut filtering approach identifies and drops non-useful cuts during each subsequent Benders iteration to reduce the size of the master problem. Non-useful cuts are cuts with no impact on the master problem feasible region.

4.3.1.1. SP Objective Proxy Prediction

In the multi-cut Benders decomposition formulation (4.2), α_ω is a variable and α^{min} is a bound where $\alpha_\omega \geq \alpha_\omega^{min}$ constitute a master problem constraint. α_ω^{min} is set as a large negative constant. The value of α_ω increases after each iteration and reaches its optimal value α_ω^* upon Benders convergence. The value of α_ω^{min} can be selected by analyzing the physical and economic aspects of the SCUC problem. Setting a suitable α_ω^{min} reduces the search space and enhances Benders convergence performance. An ideal case is to set α_ω^{min} corresponding to each subproblem ω as the optimal value of α_ω instead of using a large negative value. Fig. 4.2 shows the advantage of using optimal α_ω^*. The master problem objective reaches its optimal value in the very initial iterations. We use machine learning to predict α_ω^* for each subproblem ω by reading power demand before implementing multi-cut Benders Algorithm I.

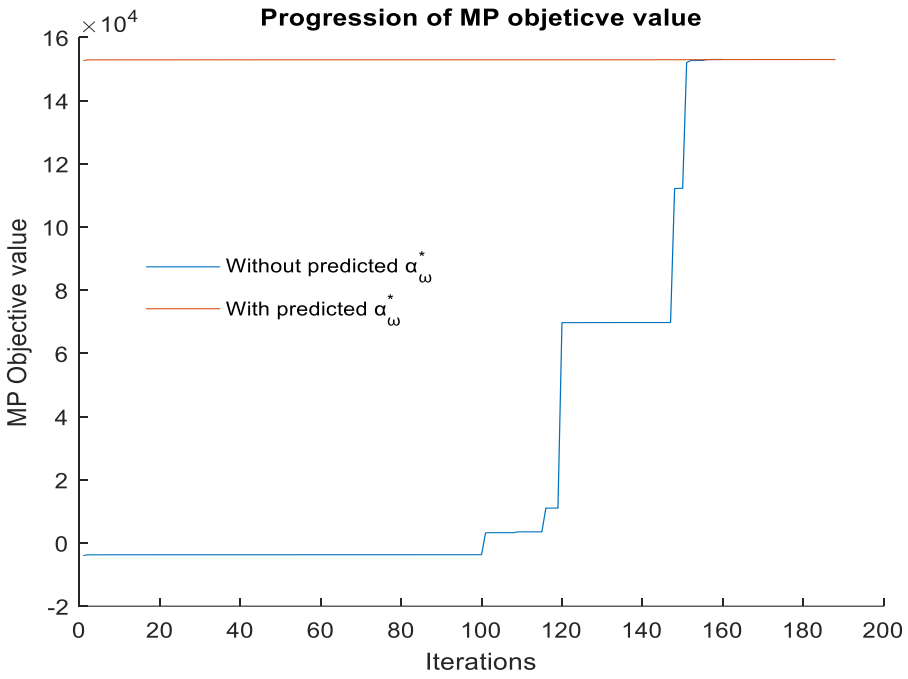


Figure 4.2. Benders convergence using α_{ω}^{*} $\forall\omega$ in inequality (4.2b).

4.3.1.2. Training Dataset Preparation

Electricity demand scenarios are input to two-stage SCUC. We have observed that α_{ω}^{*} are correlated with system demand. We use demand information as the input to a supervised learning model whose output is α_{ω}^{*}. To account for potential system operation scenarios in the training phase, we generate a set of daily load profile samples $\mathcal{D}^{s} \in \mathbb{R}^{s\times 1}$ according to (4.5). To follow the concept of demand uncertainty modeling in the two-stage SCUC problem, each $\mathcal{D}^{s}$ contains n demand profile scenarios generated by (4.6).

$$\mathcal{D}^{s} = \mathcal{D}_{base}[\eta_{s}^{L} + \eta_{s}(\eta_{s}^{U} - \eta_{s}^{L})] \qquad \forall s \qquad (4.5)$$

$$\mathcal{D}_{\omega}^{s} = \mathcal{D}^{s}[\eta_{\omega}^{L} + \eta_{\omega}(\eta_{\omega}^{U} - \eta_{\omega}^{L})] \qquad \forall\omega, \forall s \qquad (4.6)$$

where $\mathcal{D}_{\omega}^{s} \in \mathbb{R}^{s\times n}$, and $\eta_{s}(\cdot)$ and $\eta_{\omega}(\cdot)$ follows a uniform distribution between 0 and 1. Indices for samples and subsamples are denoted by s and ω. Two-stage randomness is carried out to reflect

a range of realistic operational conditions. The load point is shifted using the random parameter $\eta_s(\cdot)$ to model its daily and seasonal volatility within the range $[\eta_s^U,\ \eta_s^L]$. η_s^U and η_s^L can be determined by historical data and load growth projection or to the point when any further increase or decrease renders unit commitment infeasible. The uncertainty in subsamples is modeled using the random parameter $\eta_\omega(\cdot)$ within a specified range $[\eta_\omega^U,\ \eta_\omega^L]$. Subsamples model the hourly load randomness by multiplying $\mathcal{D}^s$ by random parameters and creating $\mathcal{D}_\omega^s$. Fig. 4.3 illustrates the load scenario generation.

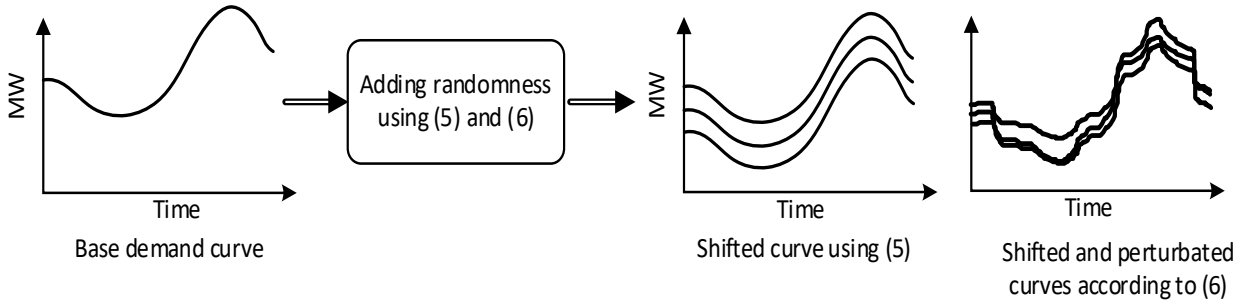


Figure 4.3. Demand scenario generation.

For a given system, Algorithm I is executed for each demand profile sample $\mathcal{D}^s$. Optimal values of proxy variables upon Benders convergence are labeled as α^{s*} and stored. Expressions (4.7) and (4.8) are, respectively, the input and target of the supervised learning model.

$$\text{input sample } s\text{: } \mathcal{D}^s = [\mathcal{D}_1^s, \mathcal{D}_2^s, \dots, \mathcal{D}_n^s]^\dagger \quad \forall s \qquad (4.7)$$

$$\text{target sample } s\text{: } \alpha^{s*} = [\alpha_1^{s*}, \alpha_2^{s*}, \alpha_3^{s*} \dots \alpha_n^{s*}]^\dagger \quad \forall s \qquad (4.8)$$

4.3.1.3. Supervised Learning Strategy

We use neural networks (NN), an efficient tool to capture the complexity and nonlinearity of a function by utilizing various activation functions. The regressor uses a fully connected NN with

mini-batch gradient descent and Rectified Linear Units (ReLU) activation functions for hidden layers. The mean squared error (MSE) is used as the loss function (9).

$$MSE = \frac{\sum_{k=1}^{K}\left(\mathcal{X}^k - \widetilde{\mathcal{X}}^k\right)^2}{\mathcal{K}} \tag{4.9}$$

Table 4.1. Hyperparameters of the learner

NN regressor	Hidden layer=1	Activation =ReLU
	Batch size=300~500	Loss function = MSE
		Optimizer = Adam

The Adam optimizer determines the optimal weight values and train the learning model. A single hidden layer is selected to have a minimalistic model. Table 4.1 shows the architecture and hyperparameters used in this study. Although more sophisticated structures could improve the results, we obtained promising outcomes with this minimalistic architecture. Fig. 4.4 illustrates a conceptual schematic of the learner model.

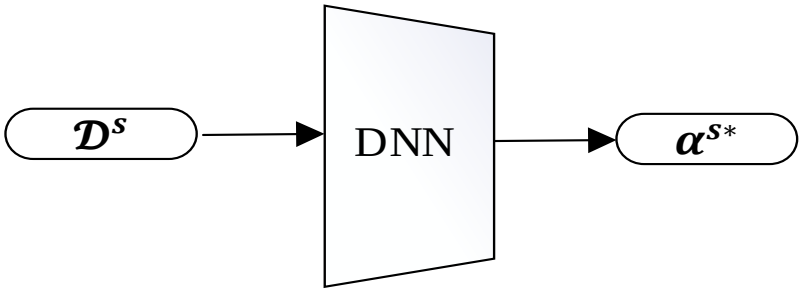


Figure 4.4. Simple representation of NN regressor reads demand profiles and predicts proxy variables.

4.3.1.4. Data Scaling

Data normalization and standardization are crucial before training a learner. The data samples are normalized by (4.10). This process enhances prediction accuracy and numerical stability.

$$\mathcal{X}_{norm} = \frac{\mathcal{X} - \mathcal{X}^{min}}{\mathcal{X}^{max} - \mathcal{X}^{min}} \tag{4.10}$$

4.3.1.5. Useful Cut Identification

The master problem still retains all cuts generated after carrying out each Benders iteration. As a result, the master problem size grows at every iteration, calling for a considerable computational memory requirement. In most cases, a subset of Benders cuts generated at each iteration contains the necessary information to build the feasible search space of the master problem. Currently, a lack of practical and systematic approach exists for classifying useful and non-useful cuts for large-scale problems [55]. Several features are suggested in [34, 36], such as cut violation, cut depth, cut order, and cut producing scenario, to identify useful cuts.

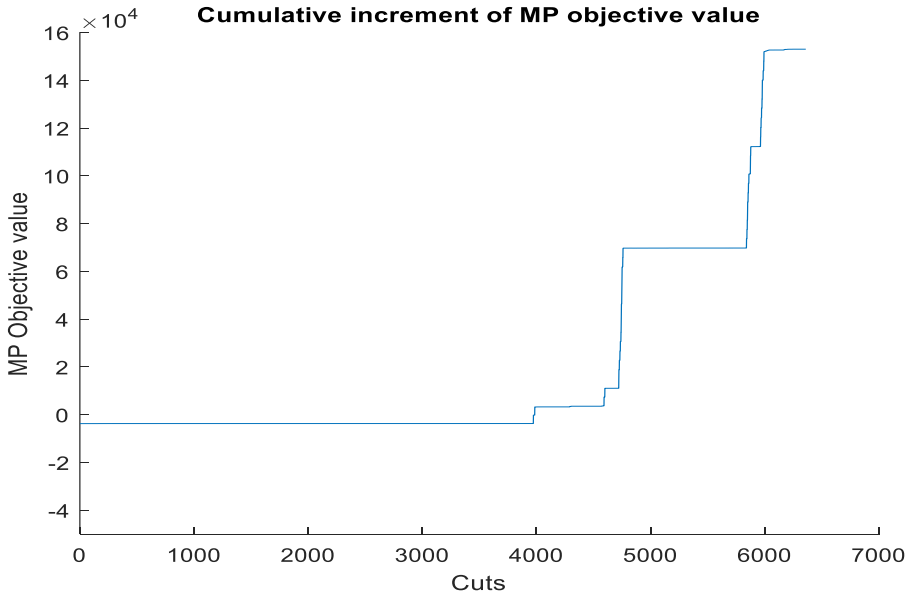


Figure 4.5. Benders lower bound improvement with cuts added cumulatively; a case of IEEE 118-bus system.

Our idea of useful cut identification is based on Fig. 4.5, which shows an example of the master problem objective value (or lower bound) improvement with cumulatively added cuts. It can be

observed that all cuts are not equally contributing to the lower bound and optimality gap improvement. Several cuts provide a positive increase in the lower bound. Such cuts can be classified as useful cuts. Other cuts do not contribute to the lower bound improvement and can be classified as non-useful. Our observation and experiment show that the numerical difference between cut values $\psi(u_{gt}^k)$ and proxy values α_ω^k can be used for useful cut identification. If an inequality constraint is satisfied as equality upon solving optimization, it is typically referred to as a binding or strictly useful constraint. Thus, if $\alpha_\omega^k - \psi(u_{gt}^k) = 0$, the cut can be identified as useful. However, many other cuts that are not satisfied as equality may also contain the necessary information to form the master problem feasible region. Such cuts should also be considered useful. As an inequality constraint approaches equality, its importance in forming a feasible region is expected to increase.

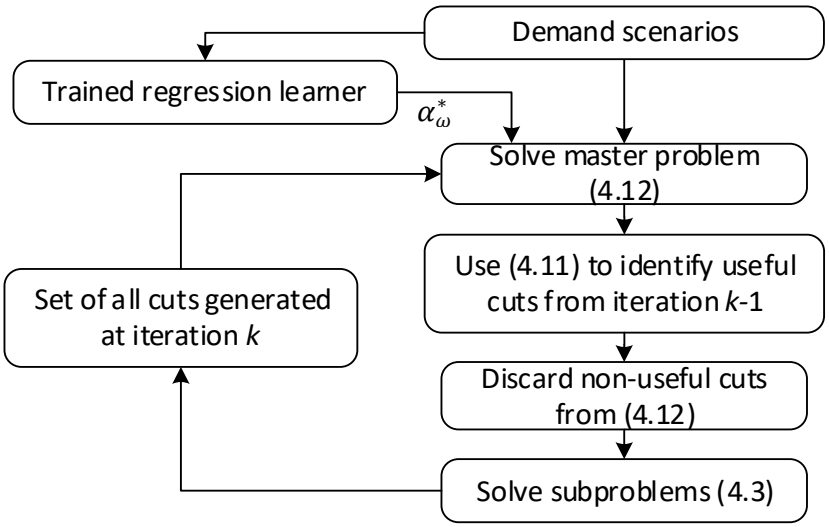


Figure 4.6. R-Benders flowchart.

Consider a demand profile sample $\mathcal{D}^s$. To capture all cuts that contain necessary information at each iteration k, we label cuts as "useful" if the difference between α_ω^k and $\psi(u_{gt}^k)$ is less than a threshold δ, selected through experimental observation. Once all cuts are generated at iteration

$k-1$, we add them to MP and solve the updated MP at iteration k to obtain u_{gt}^{k} and α_{ω}^{k}. We check (4.11) to identify useful cuts generated at iteration $k-1$. Cuts that satisfy (11) contain necessary information and are useful. We repeat this process at each iteration k to detect useful cuts generated at iteration $k-1$. Non-useful cuts are dropped from MP before moving to iteration $k+1$.

$$\hbar_{\beta}^{k-1}(u_{gt},\alpha_{\omega})^{\phi}: \left|\left|\alpha_{\omega}^{k}-\psi(u_{gt}^{k})\right|\right| \leq \delta \qquad (4.11)$$

where $\psi(u_{gt}^{k})$ is the numerical value of R.H.S of $(4.2d)$

$$\psi(u_{gt}^{k}) = J_{SP,\omega}^{k-1} + \sum_{g}\sum_{t} \lambda_{gt\omega}^{k-1}\ (u_{gt}^{k}-u_{gt}^{k-1})$$

4.3.1.6. Avoid Losing Information

Selecting a small δ results in labeling fewer cuts as useful. A large δ yields more cuts in the master problem and less computational cost reduction. Depending on the value of hyperparameter δ, we have observed cases with no cut labeled as useful at a few iterations. Retaining at least one cut from each iteration is crucial; otherwise, the proposed R-Benders may not converge due to loss of information from that particular iteration. To prevent this, we keep several cuts generated from high-load scenarios at each iteration, most of which may already be on the useful cut list. The number of retained cuts is determined through experimentation.

R-Benders MP Formulation: The R-Benders master problem at iteration k is as follows:

$$\min \sum_{\forall g}\sum_{\forall t}(SU_{g}y_{gt}+SD_{g}z_{gt}\ ) + \sum_{\forall \omega}\alpha_{\omega} \qquad (4.12a)$$

s.t

$$(1b)-(1i) \qquad (4.12b)$$

$$\cup_{i=1}^{i=k-2} \hbar_{\beta}^{n}(u_{gt}, \alpha_{\omega})^{\phi} \qquad (4.12c)$$

$$\hbar_{\beta}^{k-1}(u_{gt}, \alpha_{\omega}): \alpha_{\omega} \geq J_{SP,\omega}^{k-1} + \sum_{g}\sum_{t} \lambda_{gt\omega}^{k-1} \left(u_{gt} - u_{gt}^{*,k-1}\right) \quad \forall \omega \qquad (4.12d)$$

$$\alpha_{\omega} \geq \alpha_{\eta}\alpha_{\omega}^{*} \quad \forall \omega \qquad (4.12e)$$

where $(4.12c)$ denotes the list of all useful cuts detected until iteration $k-2$. $(4.12d)$ includes all cuts generated at iteration $k-1$. The value of α_{ω} successively progresses to reach the optimal value. To maintain the quality of the lower bound solution, we should ensure $\alpha_{\omega} \geq \alpha_{\omega}^{*}$. Given the possibility of machine learning error, the regressor predictions α_{ω}^{*} are reduced by a factor α_{η} and $\alpha_{\omega} \geq \alpha_{\omega}^{*}$ is replaced with $\alpha_{\omega} \geq \alpha_{\eta}\alpha_{\omega}^{*}$. The reduction factor α_{η} is determined by analyzing the regressor error once the training is completed and tested with new samples.

The R-Benders approach is summarized in Algorithm II and Fig. 4.6. Although a classification learning model can be trained to identify useful cuts (as in the C-Bender approach presented in Section III.B), the R-Benders approach does not use any classifier with the cost of filtering cuts with one iteration delay. The only learning model in R-Benders is a regressor to predict α_{ω}^{*}.

Algorithm II Proposed R-Benders decomposition

1. Initialize convergence tolerance ε, set iteration index $k = 0$, and set α_{η}
2. Feed predicted demand scenarios to the regressor to predict α_{ω}^{*}
3. **while** ε > tolerance
4. $\quad k = k + 1$
5. $\quad$ Solve Benders master problem (4.12) to obtain α_{ω}^{k} and u_{gt}^{k}
6. $\quad$ **If** $k \geq 2$
7. $\quad\quad$ Use Benders master problem solution obtained to identify useful cuts
8. $\quad\quad$ Drop non-useful cuts from master problem
9. $\quad$ **end if**
10. $\quad$ **for** $\boldsymbol{\omega = 1:n}$
11. $\quad\quad$ Solve SP_{ω} (3) to **obtain** $J_{SP,\omega}^{k}$ and $\lambda_{gt\omega}^{k}$
12. $\quad$ **end for**
13. $\quad$ Calculate convergence tolerance $\varepsilon = \frac{UB-LB}{abs(LB)}$

14. Form new cuts as $\alpha_\omega \geq J_{SP,\omega}^k + \sum_g \sum_t \lambda_{gt\omega}^k \left(u_{gt} - u_{gt}^k\right) \forall \omega$ and go to Step 3
15. **end while**

4.3.2. Classifier-based Cut Identification (C-Benders)

In this classifier-based approach, we use a supervised classification learner to identify useful cuts before solving the master problem at each iteration k, instead of using criterion (4.11). Once the dataset is generated, a neural network is trained with a suitable classifier loss function (e.g., F-score loss function). The model architecture and hyperparameters are similar to that of the regressor given in Table 4.1. To prepare the training dataset, all cuts must be labeled as useful or non-useful for the machine learning-based cut classifier. Once the BD algorithm converges, we collect all the cuts. To determine the label of each cut, we need to measure the effectiveness of each individual cut. We add each cut cumulatively, one after another, to the initial master problem and solve it to measure the cut's contribution to lower bound improvement. One alternative approach to labeling is forming a master problem with all cuts at first and then removing each cut one by one to see if it changes the lower bound. As opposed to the cut classification of the R-Benders approach, generating a labeled dataset of the ML-based cut classifier is computationally expensive, particularly for multi-cut Benders decomposition. The number of Benders iterations is relatively high for the SCUC problem with intertemporal constraints, leading to a multiplicatively larger number of cuts that should be individually checked. The number of cuts generated at each iteration depends on the number of demand scenarios. This is another factor that increases the computational cost of cut labeling. For instance, consider solving the 118-bus system with a 4-hour commitment horizon and a load profile sample with 40 stochastic scenarios. This case takes 289 iterations to converge with a gap of less than 1%. The total number of cuts generated for each

sample is 40×289 =11,560. To determine the label of each cut, we must solve the master problem 11,560 times.

Fig. 4.7 shows the C-Benders flowchart. The master problem is formulated in $(4.12a) - (4.12d)$, and $(4.2e)$. Filtered cuts in $(4.12c)$ are identified using the classifier. The training process is offline and needs to be performed only once. The overall accuracy of the learning-aided cut classification is slightly better than the non-learning cut classification of R-Benders.

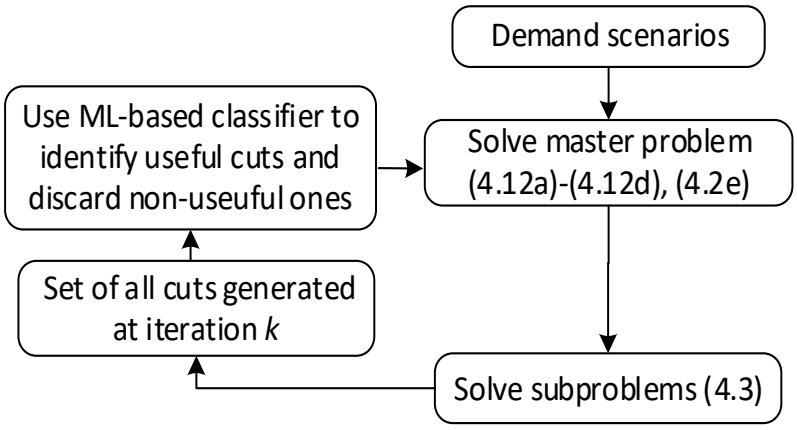


Figure 4.7. C-Benders flowchart.

4.3.3. Combined Classifier-Regressor Benders (CR-Benders)

This approach is a combination of C-Benders and R-Benders. CR-Benders uses a regressor to predict subproblem proxy variables α_ω^* and a classifier for cut filtering before starting each iteration k. This approach is C-Benders equipped with α_ω^* prediction capability.

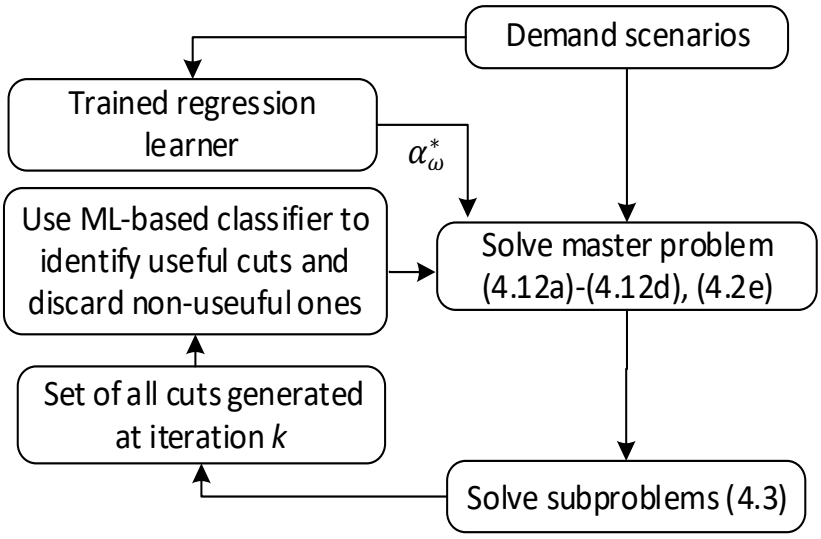


Figure 4.8. CR-Benders flowchart.

4.3.4. Subproblem Acceleration

Two non-binding constraint removal strategies can accelerate subproblem solutions, particularly for large SCUC problems. In the first learning-aided strategy, two classifiers can be trained to identify inactive (aka non-binding) network and generation ramp constraints at each Benders iteration and remove them from the model [56]. Another more straightforward, less efficient strategy is to solve a relaxed subproblem without network constraints. The relaxed subproblem solution and shift factors are used to check network constraints. The violated constraints are added to the relaxed subproblem, and the process is repeated until all network constraints are satisfied [38]. These two strategies reduce computation time and memory usage. We have used the second strategy.

4.4. Numerical Result

The effectiveness of the proposed algorithms is evaluated in comparison to conventional multi-cut Benders decomposition. The YALMIP toolkit and CPLEX are used for implementing Benders

[57], and Pytorch is used for constructing neural network models. Simulations are conducted on a computer with an Intel(R) Xeon(R) 2.10 GHz CPU and 512 GB of RAM.

4.4.1. Test Systems and Data Preparation

Three test systems with various scheduling horizons are used, including the IEEE 24-bus, and 118-bus [42]. The 24-bus system includes ten generators and 34 transmission lines. The 118-bus system consists of 54 generators and 186 lines. Each case includes 40 equiprobable load profile scenarios. Table 4.2 shows the range of load perturbation with respect to base case demand values. The convergence tolerance is set to 1% for all cases.

Table 4.2. Parameter range for demand dataset generation

System	Load	
	$[\eta_s^U, \eta_s^L]$	$[\eta_\omega^U, \eta_\omega^L]$
24-bus	70% -130%	95%-105%
118-bus	70% -130%	95%-105%

4.4.2. Useful Cut Statistics

Table 4.3 shows the average number of total cuts and useful cuts. To determine the usefulness of a cut, we have calculated the contribution of each cut to improving the objective value of the master problem. A cut is useful if it results in a non-zero increase in the objective value. The percentage of the number of useful cuts to total cuts reduces as the size of the system and the number of scheduling horizons increase. For instance, for the IEEE 118-bus system with three time periods, only 13% of cuts are labeled as useful and the rest of 87% are non-useful. Filtering non-useful cuts reduces the computational burden of the master problem significantly.

Table 4.3. Comparison of total cuts and useful cuts

Test case	Average number of total cuts	Average number of useful cuts	% of useful cuts
24-bus, 1h	224	103	46%
24-bus, 12h	1872	220	12%
118-bus, 1h	308	109	35%
118-bus, 3h	2280	288	13%

4.4.3. Master Problem Runtime and Solution Quality

Table 4.4. Average solution time (and improvement percentage) comparison in seconds

System	Benders	R-Benders	C-Benders	CR-Benders
24-bus, 1h	2.1	1.4 (33%)	0.73 (65%)	0.66 (69%)
24-bus, 12h	37.7	32.9 (13%)	9.7 (74%)	9.4 (75%)
118-bus, 1h	2.8	2.6 (7%)	2.5 (11%)	2.5 (11%)
118-bus, 3h	236	30 (87%)	57 (76%)	24 (90%)

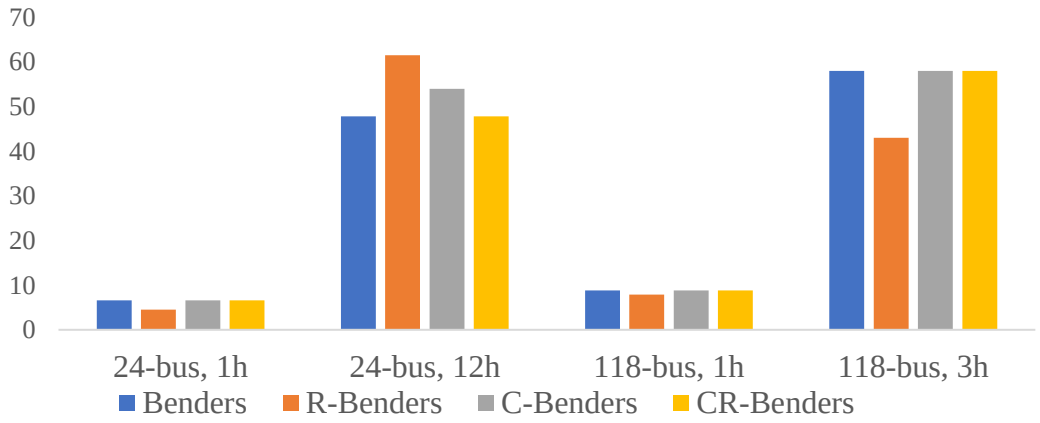


Figure 4.9. Average number of iterations.

The average solution times of conventional multi-cut Benders and the three proposed approaches are shown in Table 4.4. The improvement percentage is also reported. All three proposed approaches outperform conventional Benders. The time saving becomes more significant as the size of the optimization model increases. For instance, R-Benders reduces the master

problem solution time from 236 seconds to 30 for the 118-bus system, an 87% improvement. The time-saving of CR-Benders is more than R-Benders and C-Benders.

The average number of Benders iterations is shown in Fig. 4.9. The three proposed approaches, particularly C-Benders, and CR-Benders, take roughly the same number of iterations as conventional Benders decomposition. A comparison of the solution time and the number of iterations, as well as the number of useful cuts from Table 4.4, show that R-Benders, C-Benders, and CR-Benders can capture the necessary information to build a computationally less expensive master problem.

We use a CostGap index to assess the solution quality. The CostGap index measures the difference between the objective values generated by the proposed approaches (f^p) and the conventional Benders (f^{BD}).

$$\text{CostGap\%} = \frac{|f^p - f^{BD}|}{f^{BD}} \times 100 \qquad (4.13)$$

Table 4.5 presents the average CostGap index for all cases. The average cost gap for all test cases is negligible. For instance, the gap is less than 0.02% for the 118-bus system with a 3-hour horizon. We have observed several cases for which the proposed approaches obtain even a better solution than the conventional Benders. This is due to a warm start with stronger cuts in the first iteration.

Table 4.5. Costgap index (%)

System	R-Benders	C-Benders	CR-Benders
24-bus, 1h	0	0.13	0.07
24-bus, 12h	0.05	0.8	0.03
118-bus, 1h	0.6	0.01	0.01
118-bus, 3h	0	0.02	0.02

4.4.4. Memory Usage

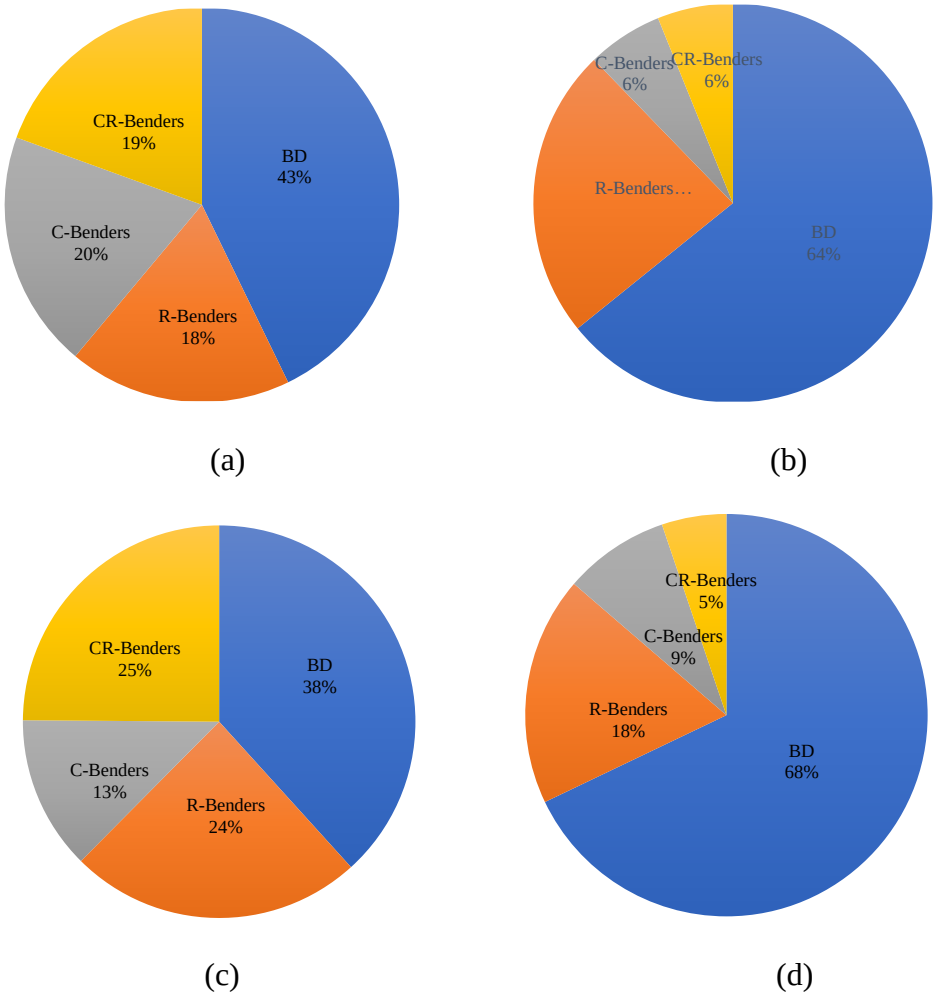


Figure 4.10. Memory usage comparison. a. 24-bus (1h) b. 24-bus (12h) b. 118-bus (1h) d. 118-bus (3h)

The memory usage of the proposed approach is compared with the conventional Benders. The memory occupied to form the constraint set from Benders cuts is illustrated in Fig. 4.10. The memory requirement reduces significantly by filtering out non-useful cuts from the model. For example, Fig. 4.10.d shows that for the 118-bus system with 3h horizon, the memory usage of CR-Benders is 87.5% less than that of the conventional Benders decomposition. Fig. 4.11 demonstrates

the number of cuts added to the master problem per iteration. The slope of the increment for the conventional Benders is steeper than that of RC-Benders.

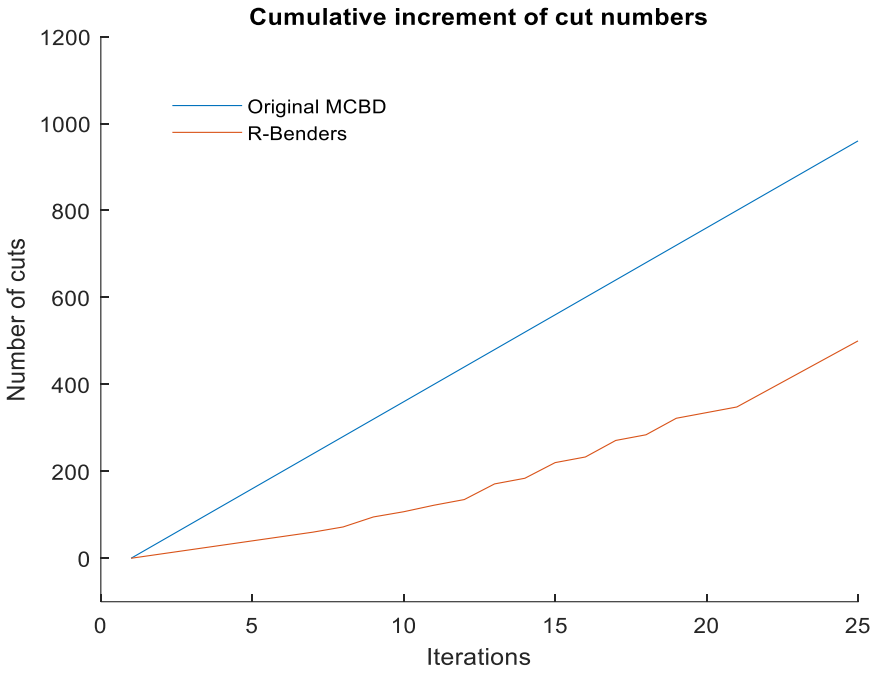


Figure 4.11. Cumulative increment of the number of cuts for IEEE 118-bus 3h case.

4.5. Conclusion

The proposed algorithms aim to reduce the computational costs of two-stage SCUC using machine learning-based warm-start and cut filtering. The proposed algorithms provide guaranteed feasible and near-optimal solutions. Conventional Benders decomposition suffers from slow convergence and might not reach an optimal point within a specified time. Not all cuts generated through Benders iterations contain useful information to form the feasible region of the master problem. Removing those non-useful cuts and forming stronger cuts enhance Benders decomposition performance. We trained a regressor to predict subproblem objective proxy variables for the master problem. The predicted values are used to form the first set of cuts in the first Benders iteration.

After solving the first iteration and onward, cuts are filtered by comparing the numerical value of cuts and the numerical value of proxy variables obtained from that particular iteration. A classifier is trained to identify useful cuts automatically after each Benders iteration. The proposed approaches are applicable to many MIP problems.

Chapter 5
Concluding Remark and Future Works

5.1. Concluding Remarks

This dissertation presents learning assisted binding constraints filtering strategy to enhance the computation efficiency of power system optimization. The utilization of machine learning (ML) is an excellent approach that can facilitate the process of analyzing data from power systems to provide meaningful insights for operators to make quick and efficient decisions. In this dissertation, three optimization functions for energy management, including AC optimal power flow (ACOPF), dynamic economic dispatch (D-ED), and security-constrained unit commitment (SCUC) are examined. Due to the computational expense and time-consuming nature of solving ACOPF for large systems, classification and regression learning models are utilized to identify inactive transmission line flow constraints and eliminate them from the optimization process to reduce computational costs. For D-ED, a learning and model-based algorithm is developed to identify the status of network and thermal unit ramp-up/down constraints, with learners reading network topology, demand, and thermal unit generation cost information as input and identifying the status of network and ramp constraints as output. Lastly, SCUC belongs to the category of mixed-integer optimization, and Benders decomposition is an effective approach for solving this class of problems. Still, it suffers from exponential worst-case computational complexity. To enhance the convergence performance of Benders decomposition, classification and regression learners are utilized to generate tighter cuts and filter out non-useful cuts at each iteration.

5.2. Future Works

There are several promising research directions for further advancing the proposed learning-aided algorithms.

- Exploring techniques to penalize false-negative classes in the learner objective function, which can reduce the risk of misclassifying
- Applying the latest computer vision and image processing techniques to analyze power system snapshots
- Applying physics-aware machine learning for electricity market clearing

Regarding Benders decomposition, several research directions can be explored, such as:

- Utilizing physics-informed learning for Benders cuts
- Formulating more meaningful indices for online useful cut identification and filtering

Quantum computing can further enhance the learning integrated Benders decomposition. Hybrid approaches combining classical and quantum computing algorithms can be explored to leverage the strengths of both paradigms, leading to even more efficient technique.

Appendix: Publication Information

IEEE COPYRIGHT FORM

To ensure uniformity of treatment among all contributors, other forms may not be substituted for this form, nor may any wording of the form be changed. This form is intended for original material submitted to the IEEE and must accompany any such material in order to be published by the IEEE. Please read the form carefully and keep a copy for your files.

Hybrid Learning Aided Inactive Constraints Filtering Algorithm to Enhance AC OPF Solution Time
Hasan, Fouad; Kargarian, Amin; Mohammadi, Javad
IEEE IAS Publications

COPYRIGHT TRANSFER

The undersigned hereby assigns to The Institute of Electrical and Electronics Engineers, Incorporated (the "IEEE") all rights under copyright that may exist in and to: (a) the Work, including any revised or expanded derivative works submitted to the IEEE by the undersigned based on the Work; and (b) any associated written or multimedia components or other enhancements accompanying the Work.

GENERAL TERMS

1. The undersigned represents that he/she has the power and authority to make and execute this form.
2. The undersigned agrees to indemnify and hold harmless the IEEE from any damage or expense that may arise in the event of a breach of any of the warranties set forth above.
3. The undersigned agrees that publication with IEEE is subject to the policies and procedures of the IEEE PSPB Operations Manual.
4. In the event the above work is not accepted and published by the IEEE or is withdrawn by the author(s) before acceptance by the IEEE, the foregoing copyright transfer shall be null and void. In this case, IEEE will retain a copy of the manuscript for internal administrative/record-keeping purposes.
5. For jointly authored Works, all joint authors should sign, or one of the authors should sign as authorized agent for the others.
6. The author hereby warrants that the Work and Presentation (collectively, the "Materials") are original and that he/she is the author of the Materials. To the extent the Materials incorporate text passages, figures, data or other material from the works of others, the author has obtained any necessary permissions. Where necessary, the author has obtained all third party permissions and consents to grant the license above and has provided copies of such permissions and consents to IEEE

BY TYPING IN YOUR FULL NAME BELOW AND CLICKING THE SUBMIT BUTTON, YOU CERTIFY THAT SUCH ACTION CONSTITUTES YOUR ELECTRONIC SIGNATURE TO THIS FORM IN ACCORDANCE WITH UNITED STATES LAW, WHICH AUTHORIZES ELECTRONIC SIGNATURE BY AUTHENTICATED REQUEST FROM A USER OVER THE INTERNET AS A VALID SUBSTITUTE FOR A WRITTEN SIGNATURE.

Amin Kargarian — **Signature**

07-01-2021 — **Date (dd-mm-yyyy)**

IEEE COPYRIGHT FORM

To ensure uniformity of treatment among all contributors, other forms may not be substituted for this form, nor may any wording of the form be changed. This form is intended for original material submitted to the IEEE and must accompany any such material in order to be published by the IEEE. Please read the form carefully and keep a copy for your files.

Topology-aware Learning Assisted Branch and Ramp Constraints Screening for Dynamic Economic Dispatch
Hasan, Fouad; Kargarian, Amin
IEEE Transactions on Power Systems

COPYRIGHT TRANSFER

The undersigned hereby assigns to The Institute of Electrical and Electronics Engineers, Incorporated (the "IEEE") all rights under copyright that may exist in and to: (a) the Work, including any revised or expanded derivative works submitted to the IEEE by the undersigned based on the Work; and (b) any associated written or multimedia components or other enhancements accompanying the Work.

GENERAL TERMS

1. The undersigned represents that he/she has the power and authority to make and execute this form.
2. The undersigned agrees to indemnify and hold harmless the IEEE from any damage or expense that may arise in the event of a breach of any of the warranties set forth above.
3. The undersigned agrees that publication with IEEE is subject to the policies and procedures of the IEEE PSPB Operations Manual.
4. In the event the above work is not accepted and published by the IEEE or is withdrawn by the author(s) before acceptance by the IEEE, the foregoing copyright transfer shall be null and void. In this case, IEEE will retain a copy of the manuscript for internal administrative/record-keeping purposes.
5. For jointly authored Works, all joint authors should sign, or one of the authors should sign as authorized agent for the others.
6. The author hereby warrants that the Work and Presentation (collectively, the "Materials") are original and that he/she is the author of the Materials. To the extent the Materials incorporate text passages, figures, data or other material from the works of others, the author has obtained any necessary permissions. Where necessary, the author has obtained all third party permissions and consents to grant the license above and has provided copies of such permissions and consents to IEEE

BY TYPING IN YOUR FULL NAME BELOW AND CLICKING THE SUBMIT BUTTON, YOU CERTIFY THAT SUCH ACTION CONSTITUTES YOUR ELECTRONIC SIGNATURE TO THIS FORM IN ACCORDANCE WITH UNITED STATES LAW, WHICH AUTHORIZES ELECTRONIC SIGNATURE BY AUTHENTICATED REQUEST FROM A USER OVER THE INTERNET AS A VALID SUBSTITUTE FOR A WRITTEN SIGNATURE.

Amin Kargarian — **Signature**

11-01-2022 — **Date (dd-mm-yyyy)**

Information for Authors

AUTHOR RESPONSIBILITIES

The IEEE distributes its technical publications throughout the world and wants to ensure that the material submitted to its publications is properly available to the readership of those publications. Authors must ensure that their Work meets the requirements as stated in section 8.2.1 of the IEEE PSPB Operations Manual, including provisions covering originality,

---

authorship, author responsibilities and author misconduct. More information on IEEE's publishing policies may be found at http://www.ieee.org/publications_standards/publications/rights/authorrightsresponsibilities.html Authors are advised especially of IEEE PSPB Operations Manual section 8.2.1.B12: "It is the responsibility of the authors, not the IEEE, to determine whether disclosure of their material requires the prior consent of other parties and, if so, to obtain it." Authors are also advised of IEEE PSPB Operations Manual section 8.1.1B: "Statements and opinions given in work published by the IEEE are the expression of the authors."

RETAINED RIGHTS/TERMS AND CONDITIONS

- Authors/employers retain all proprietary rights in any process, procedure, or article of manufacture described in the Work.
- Authors/employers may reproduce or authorize others to reproduce the Work, material extracted verbatim from the Work, or derivative works for the author's personal use or for company use, provided that the source and the IEEE copyright notice are indicated, the copies are not used in any way that implies IEEE endorsement of a product or service of any employer, and the copies themselves are not offered for sale.
- Although authors are permitted to re-use all or portions of the Work in other works, this does not include granting third-party requests for reprinting, republishing, or other types of re-use.The IEEE Intellectual Property Rights office must handle all such third-party requests.
- Authors whose work was performed under a grant from a government funding agency are free to fulfill any deposit mandates from that funding agency.

AUTHOR ONLINE USE

- **Personal Servers**. Authors and/or their employers shall have the right to post the accepted version of IEEE-copyrighted articles on their own personal servers or the servers of their institutions or employers without permission from IEEE, provided that the posted version includes a prominently displayed IEEE copyright notice and, when published, a full citation to the original IEEE publication, including a link to the article abstract in IEEE Xplore. Authors shall not post the final, published versions of their papers.
- **Classroom or Internal Training Use.** An author is expressly permitted to post any portion of the accepted version of his/her own IEEE-copyrighted articles on the author's personal web site or the servers of the author's institution or company in connection with the author's teaching, training, or work responsibilities, provided that the appropriate copyright, credit, and reuse notices appear prominently with the posted material. Examples of permitted uses are lecture materials, course packs, e-reserves, conference presentations, or in-house training courses.
- **Electronic Preprints.** Before submitting an article to an IEEE publication, authors frequently post their manuscripts to their own web site, their employer's site, or to another server that invites constructive comment from colleagues. Upon submission of an article to IEEE, an author is required to transfer copyright in the article to IEEE, and the author must update any previously posted version of the article with a prominently displayed IEEE copyright notice. Upon publication of an article by the IEEE, the author must replace any previously posted electronic versions of the article with either (1) the full citation to the IEEE work with a Digital Object Identifier (DOI) or link to the article abstract in IEEE Xplore, or (2) the accepted version only (not the IEEE-published version), including the IEEE copyright notice and full citation, with a link to the final, published article in IEEE Xplore.

Questions about the submission of the form or manuscript must be sent to the publication's editor.
Please direct all questions about IEEE copyright policy to:
IEEE Intellectual Property Rights Office, copyrights@ieee.org, +1-732-562-3966

References

[1] J. Carpentier, "Contribution a l'etude du dispatching economique," *Bulletin de la Societe Francaise des Electriciens,* vol. 3, no. 1, pp. 431-447, 1962.

[2] J. Carpentier, "Optimal power flows: uses, methods and developments," *IFAC Proceedings Volumes,* vol. 18, no. 7, pp. 11-21, 1985.

[3] H. Happ, "Optimal power dispatchꟾA comprehensive survey," *IEEE Transactions on Power Apparatus and Systems,* vol. 96, no. 3, pp. 841-854, 1977.

[4] M. Huneault and F. Galiana, "A survey of the optimal power flow literature," *IEEE transactions on Power Systems,* vol. 6, no. 2, pp. 762-770, 1991.

[5] M. B. Cain, R. P. O'neill, and A. Castillo, "History of Optimal Power Flow and Formulations Optimal Power Flow Paper 1," 2012.

[6] S. H. Low, "Convex relaxation of optimal power flow—Part I: Formulations and equivalence," *IEEE Transactions on Control of Network Systems,* vol. 1, no. 1, pp. 15-27, 2014.

[7] S. H. Low, "Convex relaxation of optimal power flow—Part II: Exactness," *IEEE Transactions on Control of Network Systems,* vol. 1, no. 2, pp. 177-189, 2014.

[8] 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.

[9] 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, 2016.

[10] D. K. Molzahn and I. A. Hiskens, "A survey of relaxations and approximations of the power flow equations," *Foundations and Trends® in Electric Energy Systems,* vol. 4, no. 1-2, pp. 1-221, 2019.

[11] K. Baker, "Learning warm-start points for AC optimal power flow," in *2019 IEEE 29th International Workshop on Machine Learning for Signal Processing (MLSP)*, 2019: IEEE, pp. 1-6.

[12] F. Fioretto, T. W. Mak, and P. Van Hentenryck, "Predicting AC optimal power flows: Combining deep learning and lagrangian dual methods," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020, vol. 34, no. 01, pp. 630-637.

[13] M. Chatzos, F. Fioretto, T. W. Mak, and P. Van Hentenryck, "High-Fidelity Machine Learning Approximations of Large-Scale Optimal Power Flow," *arXiv preprint arXiv:2006.16356,* 2020.

[14] A. Velloso and P. Van Hentenryck, "Combining Deep Learning and Optimization for Security-Constrained Optimal Power Flow," *arXiv preprint arXiv:2007.07002,* 2020.

[15] X. Pan, M. Chen, T. Zhao, and S. H. Low, "DeepOPF: A Feasibility-Optimized Deep Neural Network Approach for AC Optimal Power Flow Problems," *arXiv preprint arXiv:2007.01002,* 2020.

[16] A. S. Zamzam and K. Baker, "Learning optimal solutions for extremely fast AC optimal power flow," in *2020 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm),* 2020: IEEE, pp. 1-6.

[17] Y. Yang, Z. Yang, J. Yu, K. Xie, and L. Jin, "Fast Economic Dispatch in Smart Grids Using Deep Learning: An Active Constraint Screening Approach," *IEEE Internet of Things Journal,* 2020.

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

[19] K. Baker, "A Learning-boosted Quasi-Newton Method for AC Optimal Power Flow," *arXiv preprint arXiv:2007.06074,* 2020.

[20] Á. S. Xavier, F. Qiu, and S. Ahmed, "Learning to solve large-scale security-constrained unit commitment problems," *INFORMS Journal on Computing,* 2020.

[21] S. Pineda, J. M. Morales, and A. Jimenez-Cordero, "Data-Driven Screening of Network Constraints for Unit Commitment," *IEEE Transactions on Power Systems,* 2020.

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

[23] 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.

[24] A. J. Ardakani and F. Bouffard, "Prediction of Umbrella Constraints," in *IEEE Power Systems Computation Conference (PSCC)*, 2018, pp. 1-7.

[25] K. Baker and A. Bernstein, "Joint Chance Constraints in AC Optimal Power Flow: Improving Bounds through Learning," *IEEE Transactions on Smart Grid,* 2019.

[26] J. BnnoBRs, "Partitioning procedures for solving mixed-variables programming problems ‘," *Numerische mathematik,* vol. 4, no. 1, pp. 238-252, 1962.

[27] M. Minoux, *Mathematical programming: theory and algorithms*. John Wiley & Sons, 1986.

[28] K. Holmberg, "On the convergence of cross decomposition," *Mathematical Programming,* vol. 47, no. 1, pp. 269-296, 1990.

[29] A. I. Corréa, A. Langevin, and L.-M. Rousseau, "Scheduling and routing of automated guided vehicles: A hybrid approach," *Computers & operations research,* vol. 34, no. 6, pp. 1688-1707, 2007.

[30] J.-F. Cordeau, G. Stojković, F. Soumis, and J. Desrosiers, "Benders decomposition for simultaneous aircraft routing and crew scheduling," *Transportation science,* vol. 35, no. 4, pp. 375-388, 2001.

[31] M. I. Restrepo Ruiz, "Grammar-Based Decomposition Methods for Multi-Activity Tour Scheduling," École Polytechnique de Montréal, 2015.

[32] J. Naoum-Sawaya and S. Elhedhli, "A nested Benders decomposition approach for telecommunication network planning," *Naval Research Logistics (NRL),* vol. 57, no. 6, pp. 519-539, 2010.

[33] K. Brandes, "Implementierung und analyse verschiedener strategien zur aggregation und disaggregation von multi-cuts im benders dekompositionsverfahren," Master’s thesis, Universität Paderborn, North Rhine-Westphalia, Germany, 2011.

[34] H. Jia and S. Shen, "Benders cut classification via support vector machines for solving two-stage stochastic programs," *INFORMS Journal on Optimization,* vol. 3, no. 3, pp. 278-297, 2021.

[35] B. Vandenbussche, S. Delikaraoglou, I. Blanco, and G. Hug, "Data-driven adaptive benders decomposition for the stochastic unit commitment problem," *arXiv preprint arXiv:1912.01039,* 2019.

[36] M. Lee, N. Ma, G. Yu, and H. Dai, "Accelerating generalized benders decomposition for wireless resource allocation," *IEEE Transactions on Wireless Communications,* vol. 20, no. 2, pp. 1233-1247, 2020.

[37] A. Ruszczyński, "Decomposition methods," *Handbooks in operations research and management science,* vol. 10, pp. 141-211, 2003.

[38] D. A. Tejada-Arango, P. Sánchez-Martın, and A. Ramos, "Security constrained unit commitment using line outage distribution factors," *IEEE Transactions on power systems,* vol. 33, no. 1, pp. 329-337, 2017.

[39] F. Hasan and A. Kargarian, "Topology-aware Learning Assisted Branch and Ramp Constraints Screening for Dynamic Economic Dispatch," *IEEE Transactions on Power Systems*, 2022.

[40] R. D. Zimmerman, C. E. Murillo-Sánchez, and R. J. Thomas, "MATPOWER: Steady-state operations, planning, and analysis tools for power systems research and education," *IEEE Transactions on power systems*, vol. 26, no. 1, pp. 12-19, 2011.

[41] P. Branco, L. Torgo, and R. Ribeiro, "A survey of predictive modelling under imbalanced distributions," *arXiv preprint arXiv:1505.01658*, 2015.

[42] S. Babaeinejadsarookolaee *et al.*, "The power grid library for benchmarking ac optimal power flow algorithms," *arXiv preprint arXiv:1908.02788*, 2019.

[43] Y. Chen, Y. Wang, D. Kirschen, and B. J. I. T. o. P. S. Zhang, "Model-free renewable scenario generation using generative adversarial networks," vol. 33, no. 3, pp. 3265-3275, 2018.

[44] Y. Chen, X. Wang, and B. Zhang, "An unsupervised deep learning approach for scenario forecasts," in *2018 Power Systems Computation Conference (PSCC)*, 2018: IEEE, pp. 1-7.

[45] I. Goodfellow *et al.*, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672-2680.

[46] D. W. Ross and S. Kim, "Dynamic economic dispatch of generation," *IEEE transactions on power apparatus and systems*, no. 6, pp. 2060-2068, 1980.

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

[48] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *International Conference on Machine Learning*, 2019: PMLR, pp. 6105-6114.

[49] J. Lofberg, "YALMIP: A toolbox for modeling and optimization in MATLAB," in *2004 IEEE international conference on robotics and automation (IEEE Cat. No. 04CH37508)*, 2004: IEEE, pp. 284-289.

[50] A. J. Conejo and L. Baringo, *Power system operations*. Springer, 2018.

[51] A. J. Conejo, E. Castillo, R. Minguez, and R. Garcia-Bertrand, *Decomposition techniques in mathematical programming: engineering and science applications*. Springer Science & Business Media, 2006.

[52] J. R. Birge and F. V. Louveaux, "A multicut algorithm for two-stage stochastic linear programs," *European Journal of Operational Research*, vol. 34, no. 3, pp. 384-392, 1988.

[53] L. Su, L. Tang, and I. E. Grossmann, "Computational strategies for improved MINLP algorithms," *Computers & Chemical Engineering,* vol. 75, pp. 40-48, 2015.

[54] A. Nasri, S. J. Kazempour, A. J. Conejo, and M. Ghandhari, "Network-constrained AC unit commitment under uncertainty: A Benders' decomposition approach," *IEEE transactions on power systems,* vol. 31, no. 1, pp. 412-422, 2015.

[55] R. Rahmaniani, T. G. Crainic, M. Gendreau, and W. Rei, "The Benders decomposition algorithm: A literature review," *European Journal of Operational Research,* vol. 259, no. 3, pp. 801-817, 2017.

[56] F. Hasan and A. Kargarian, "Topology-Aware Learning Assisted Branch and Ramp Constraints Screening for Dynamic Economic Dispatch," *IEEE Transactions on Power Systems,* vol. 37, no. 5, pp. 3495-3505, 2022.

[57] J. Lfberg, "A toolbox for modeling and optimization in MATLAB," in *Proceedings of the Conference on Computer-Aided Control System Design (CACSD) p,* 2004, vol. 284289.

VITA

Fouad Hasan received his B.Sc. degree in Electrical and Electronic Engineering from BUET Bangladesh, in 2015. He has completed his M.Sc. with the Department of Electrical and Computer Engineering, Louisiana State University, LA, USA, and continued his Ph.D. degree in same department, working at the interface of energy, optimization and machine learning. His research interests focus on the interface of Mathematical Optimization and Machine Learning. His primary research purpose is to develop computationally efficient and scalable algorithms for large-scale optimization problems in power systems.

Publications:

Journal Papers

- **F. Hasan**, A. Kargarian, J. Mohammadi, "Hybrid Learning Aided Inactive Constraints Filtering Algorithm to Enhance ACOPF Solution Time" **IEEE Transactions on Industry Applications** (Volume: 57, Issue: 2, March-April 2021)
- **F. Hasan**, A. Kargarian, "Topology-Aware Learning Assisted Branch and Ramp Constraints Screening for Dynamic Economic Dispatch" **IEEE Transactions on Power Systems** (Volume: 37, Issue: 5, September 2022)
- F. Safdarian, A. Kargarian, **F. Hasan**, "Multiclass Learning-Based Temporal Decomposition and Distributed Optimization for Power Systems" **IEEE Transactions on Power Systems** (Volume: 36, Issue: 6, November 2021)
- C. Wu**, F. Hasan**, A. Kargarian, "Scalable Distributionally Robust Joint Chance-Constrained Unit Commitment" **IEEE Transaction on Power Systems** (under review)

Draft Completed

- **F. Hasan**, A. Kargarian, " Accelerating L-Shaped Two-Stage Stochastic SCUC With Learning Integrated Benders Decomposition " **IEEE Transaction on Power Systems** (Draft completed and will be submitted)

Conference Papers

- Hasan, A. Kargarian, and A. Mohammadi, "A survey on applications of machine learning for optimal power flow," in 2020 **IEEE Texas Power and Energy Conference** (TPEC), 2020: IEEE, pp. 1-6.

- Hasan and A. Kargarian, "Combined learning and analytical model based early warning algorithm for real-time congestion management," in 2020 **IEEE Texas Power and Energy Conference** (TPEC), 2020: IEEE, pp. 1-6.