Combining Deep Learning and Optimization for Preventive Security-Constrained DC Optimal Power Flow

Alexandre Velloso and Pascal Van Hentenryck, *Member, IEEE*

***Abstract*—The security-constrained optimal power flow (SCOPF) is fundamental in power systems and connects the automatic primary response (APR) of synchronized generators with the short-term schedule. Every day, the SCOPF problem is repeatedly solved for various inputs to determine robust schedules given a set of contingencies. Unfortunately, the modeling of APR within the SCOPF problem results in complex large-scale mixed-integer programs, which are hard to solve. To address this challenge, leveraging the wealth of available historical data, this paper proposes a novel approach that combines deep learning and robust optimization techniques. Unlike recent machine-learning applications where the aim is to mitigate the computational burden of exact solvers, the proposed method predicts directly the SCOPF implementable solution. Feasibility is enforced in two steps. First, during training, a Lagrangian dual method penalizes violations of physical and operations constraints, which are iteratively added as necessary to the machine-learning model by a Column-and-Constraint-Generation Algorithm (CCGA). Second, another different CCGA restores feasibility by finding the closest feasible solution to the prediction. Experiments on large test cases show that the method results in significant time reduction for obtaining feasible solutions with an optimality gap below 0.1%.**

***Index Terms*—Column and constraint generation, decomposition methods, deep learning, neural network, primary response, security-constrained optimal power flow.**

NOMENCLATURE

This section introduces the main notation. Bold symbols are used for matrices (uppercase) and vectors (lowercase). Additional symbols are either explained in the context or interpretable by applying the following general rules: Symbols with superscript "j," or "k" denote new variables, parameters or sets corresponding to the j-th, or k-th iteration of the associated method. Symbols with superscript "$*$" denote the optimal value of the associated (iterating) variable. Symbols with superscript "t'" are associated with the data set for the t-th past solve. Dotted symbols are associated with predictors for the corresponding variable.

Manuscript received July 13, 2020; revised December 1, 2020; accepted January 9, 2021. Date of publication January 25, 2021; date of current version June 18, 2021. This work is partially supported by NSF Grant NSF-2007095. Paper no. TPWRS-01174-2020. *(Corresponding author: Alexandre Velloso.)*

The authors are with the School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, GA 30332 USA (e-mail: alevelloso@gmail.com; pascal.vanhentenryck@isye.gatech.edu).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TPWRS.2021.3054341.

Digital Object Identifier 10.1109/TPWRS.2021.3054341

Sets

$\mathcal{E}, \mathcal{E}_s$	Feasibility sets for variables associated with the nominal state and contingent state s, respectively.
$\mathcal{F}_s$	Feasibility set for primary response variables under contingent state s.
$\mathcal{C}, \mathbb{C}$	Full set and subset of constraints.
$\mathcal{G}, \mathcal{L}, \mathcal{N}$	Sets of generators, transmission lines and buses, respectively.
$\mathcal{S}, \mathbb{S}$	Full set and subset of contingencies, respectively.
$\mathcal{T}, \mathbb{T}$	Full set and subset of past solves, respectively.
$\mathbb{U}^+, \mathbb{U}^-$	Subsets of line–contingent state pairs.
$\mathcal{Y}_s$	Set of decision variables associated with automatic primary response under contingent state s.

Parameters

α, ρ	Learning rate and Lagragian dual step size.
β, β_1, β_c	Parameters for selecting constraints.
$\boldsymbol{\gamma}$	Vector of parameters for primary response.
γ_i	Parameter for primary response of generator i.
ϵ	Tolerance for transmission line violation.
$\boldsymbol{\lambda}, \lambda_c$	Vectors for all Lagrangian multipliers and Lagrangian multipliers for constraint c.
$\boldsymbol{\nu}, \nu_c, \tilde{\nu}_c$	Vector for violations, violation for constraint c, and median violation for c among past solves $\mathcal{T}$.
$\mathbf{A}, \mathbf{B}$	Line-bus and Generator-bus incidence matrices.
$\boldsymbol{\omega}$	Vector of weights for deep neural network.
$\mathbf{d}$	Vector of nodal net loads.
$\mathbf{e}$	Vector of ones with appropriate dimension.
$\overline{\mathbf{f}}$	Vector of line capacities.
$\underline{\mathbf{g}}, \overline{\mathbf{g}}$	Vectors of lower and upper limits for generators.
$\overline{g}_i$	Upper limit for generator i.
$\hat{\mathbf{g}}$	Vector of capacities for generators.
$\hat{g}_i$	Capacity of generator i.
$h(\cdot)$	Piecewise linear generation costs.
$\mathbf{K}_0$	Matrix of power transfer distribution factors.
$\mathbf{K}_1$	Preprocessed matrix for flow limits.
$\mathbf{k}_2, \mathbf{k}_3$	Preprocessed vectors for flow limits.
$\overline{\mathbf{r}}$	Vector of primary response limits of generators.
$\overline{r}_i$	Element of $\overline{\mathbf{r}}$ related to generator i, given by $\gamma_i \hat{g}_i$.
$\mathbf{S}$	Angle-to-flow matrix.

Nominal-state-related decision variables and vectors

$\boldsymbol{\theta}, \mathbf{f}, \mathbf{g}$	Phase angles, line flows, and nominal generation.
g_i	Generation of generator i in nominal state.

Contingent-state-related decision variables and vectors

$\boldsymbol{\theta}_s$	Vector of phase angles under contingent state s.
$\boldsymbol{\tau}_s^+, \boldsymbol{\tau}_s^-$	Vectors of line violation under contingent state s.
ϕ, s_ϕ	Highest line violation and related contingent state.
$\tilde{\phi}$	Median highest line violation among instances $\mathcal{T}$.
$\mathbf{f}_s$	Vector for line flows under contingent state s.
$\mathbf{g}_s$	Vector for generation under contingent state s.
$\mathbf{g}'_s$	Provisional vector for $\mathbf{g}_s$.
$g_{s,i}$	Generation of generator i under contingent state s.
$g'_{s,i}$	Provisional variable for $g_{s,i}$.
n_s	Global signal under contingent state s.
$\mathbf{x}_s$	Binary vector indicating whether generators reached $\overline{\mathbf{g}}$ under contingency state s.
$x_{s,i}$	Element of $\mathbf{x}_s$ corresponding to generator i.

I. Introduction

A. Motivation

POWER systems operations require constant equilibrium between nodal loads and generation. At the scale of seconds, this balance is achieved by *Automatic Primary Response* (APR) mechanisms that govern the *synchronized generators*. For longer time scales, ranging from a few minutes to hours or even days ahead, this balance is obtained by solving mathematical optimization problems, as independent system operators seek consistent and efficient schedules satisfying complex physical and operational constraints. The need to solve these optimization problems in a timely manner is driving intense research about new models and algorithms, both in industry and academia. In this vein, this work aims at speeding up solution times of security-constrained optimal power flow (SCOPF) problem [1]–[7] by combining deep learning and robust optimization methods. The SCOPF is solved by operators every few minutes for different sets of bus loads. The high penetration of renewable sources of energy has increased the frequency of these optimizations. The SCOPF problem considered in this work links the APR to the very short-term schedule. It is also relevant to mention that the SCOPF problem is directly or indirectly present in many other power system applications, including security-constrained unit commitment [8], transmission switching [9], and expansion planning [10]. Thus, a reduction in the computational burden would allow system operators to introduce important modeling improvements to many applications.

B. Contextualization and Related Work

The SCOPF problem determines a least-cost pre-contingency generator dispatch that allows for feasible points of operation for a set of *contingencies*, e.g., individual failures of main lines and/or generators. The SCOPF problem may refer to the *corrective* case [5] where re-scheduling is deemed possible and to the *preventive* case where no re-dispatch occurs [3], [6], [11], i.e., the system must be able to achieve a feasible steady-state point without a new schedule. A valuable review of the SCOPF problem and solution methods is available in [4]. Interesting discussions about credible contingencies, reserve requirements, security criteria, and regulation for reserves can be found in [2], [6], [12] and the references therein. Without loss of generality, the $N-1$ security criterion for generators is adopted in this paper, i.e., the system must operate under the loss of any single generator.

The SCOPF is a nonlinear and nonconvex problem based on the AC optimal power flow (OPF) equations. Extensive reviews can be found in [13] and [14]. The DC formulation of the SCOPF has been widely used in both academia and industry [2], [3], [6], [7]. Interesting discussions regarding the quality of approximations and relaxations of the OPF problem can be found in [15]–[18]. The DC-SCOPF can also be used to improve AC-SCOPF approaches [19]. It is not within the scope of this work to discuss the quality of aforementioned approximations or relaxations to the optimal power flow. Instead, this research offers a new approach that improves current industry practices, which is still strongly based on DC-SCOPF.

The APR of synchronized generators is essential for stability. These generators respond automatically to frequency variations, caused by power imbalances for instance, by adjusting their power outputs until frequency is normalized and the power balance is restored. Unfortunately, the APR deployment, which is bounded by generators limits only, may result in transmission line overloads [6], [11], [20]. Therefore, this work co-optimizes the APR of synchronized generators within the (preventive) SCOPF problem. Even though the APR behavior is nonlinear, linear approximations are used in practice [21]. In [6], [7], [11], [22], [23], the APR is modeled by a single variable representing frequency drop (or power loss) for each contingency state and by a participation factor for each generator.

The preventive DC-SCOPF problem with APR is referred to as the SCOPF problem for conciseness in this paper. It admits an exact *extensive formulation* (with all variables and constraints for nominal state and contingency states) as a mixed-integer linear program (MILP) [6], [11]. Nevertheless, this formulation is generally very large because the APR constraints require binary variables for each generator and for each contingency state to determine whether generators are producing according to the linear response model or are at their limits [6], [11]. Thus, the number of binary variables increases quadratically with the number of generators, which makes the extensive form of the SCOPF impractical. Better modeling strategies and decomposition schemes are required.

The robust optimization framework has been widely applied in power systems due to its interesting tradeoff between modeling capability and tractability. See [24] for a review of robust optimization applications in power systems. The SCOPF problem can be modeled as a two-stage robust optimization or adaptive robust optimization (ARO) and tackled by two main decomposition methods: Benders decomposition [25] and the column-and-constraint-generation algorithm (CCGA) [26]. Both approaches rely on iterative procedures that solve a *master problem* and *subproblems*. The master problem is basically the nominal OPF problem with additional cuts/constraints and variables representing feasibility or optimality information on the subproblems. Whereas Benders decomposition provides dual information about the subproblems through valid cuts restricting

the master problem, the CCGA adds primal contraints and variables from the subproblems to the master problem.

Unfortunately, the SCOPF problem is not suitable for a traditional Benders decomposition since the subproblems are non-convex due to the APR constraints (which feature binary variables). Despite such challenges, inspired by [25], an interesting heuristic method was proposed in [6] but it does not guarantee optimality. In contrast, the CCGA algorithm proposed in [11] is an exact solution method which was used to produce optimal solutions to power network with more than 2000 buses. Notwithstanding aforementioned contributions, both approaches still require significant computational effort to obtain near-optimal solutions.

Machine learning (ML) approaches have been advocated to address the computational burden associated with the hard and repetitive optimization problems in the power sector, given the large amounts of historical data (i.e., past solutions). Initial attempts date back to the early 1990s, when, for example, artificial neural networks were applied to predict the on/off decisions of generators for an unit commitment problem [27], [28]. More recently, ML was used to identify partial warm-start solutions and/or constraints that can be omitted, and to determine affine subspaces where the optimal solution is likely to lie [29]. Artificial neural network and decision tree regression were also used to learn sets of high-priority lines to consider for transmission switching [30], while the k-nearest neighbors approach was used to select previously optimized topologies directly from data [31]. As for the security and reliability aspects of the network, the security-boundary detection was modeled with a neural network to simplify stability constraints for the optimal power flow [32], while decision trees were applied to determine security boundaries (regions) for controllable variables for a coupled natural gas and electricity system [33]. Machine learning was also applied for identifying the relevant sets of active constraints for the OPF problem [34].

Unlike these applications, where the main purpose of machine learning is to enhance the solver performance by classifying sets, eliminating constraints, and/or by modeling specific parts of the problems, the machine-learning approach in [35] directly predicts the generator dispatch for the OPF by combining deep learning and Lagrangian duality. This approach produces significant computational gains but is not directly applicable to the SCOPF problem which features an impractical number of variables and constraints. This work remedies this limitation.

C. Contributions

The paper assumes the existence of historical SCOPF data, i.e., pairs of inputs and outputs [29]–[31], [34], [35]. The proposed approach uses a deep neural network (DNN) to approximate the mapping between loads and optimal generator dispatches. To capture the physical, operational, and APR constraints, the paper applies the Lagrangian dual scheme of [35] that penalizes constraint violations at training time. Moreover, to ensure computational tractability, the training process, labeled as CCGA-DNN, mimics a dedicated CCGA algorithm that iteratively adds new constraints for a few critical contingencies. In these constraints, an approximation for the post-contingency generation is adopted to keep the size of the DNN small. The resulting DNN provides high-quality approximations to the SCOPF in milliseconds and can be used to seed another dedicated CCGA to find the nearest feasible solution to the prediction. The resulting approach may bring two orders of magnitude improvement in efficiency compared to the original CCGA algorithm.

In summary, the contributions can be summarized as follows: i) a novel DNN that maps a load profile onto a high-quality approximation of the SCOPF problem, ii) a new training procedure, the CCGA-DNN, that mimics a CCGA, where the master optimization problem is replaced by a DNN prediction, iii) an approximation for the post-contingency generation which keeps the DNN size small, and iv) a dedicated CCGA algorithm seeded with the DNN evaluation to obtain high-quality feasible solutions fast. Of particular interest is the tight combination of machine learning and optimization proposed by the approach.

D. Paper Organization

This paper is organized as follows. Section II introduces the SCOPF problem. Section III presents the properties of the SCOPF problem and the CCGA for SCOPF. Section IV introduces the deep learning models in stepwise refinements. Section V describes the CCGA for feasibility recovery. Section VI reports the case studies and the numerical experiments, and Section VII discusses extensions and limitations. Finally, Section VIII concludes the paper.

II. The SCOPF Problem

A. The Power Flow Constraints

The SCOPF formulation uses traditional security-constrained DC power flow constraints over the vectors for generation $\mathbf{g}$, flows $\mathbf{f}$, and phase angles $\boldsymbol{\theta}$. In matrix notations, these constraints are represented as follows:

$$\mathbf{A}\mathbf{f} + \mathbf{B}\mathbf{g} = \mathbf{d} \tag{1}$$

$$\mathbf{f} = \mathbf{S}\boldsymbol{\theta} \tag{2}$$

$$-\overline{\mathbf{f}} \leq \mathbf{f} \leq \overline{\mathbf{f}} \tag{3}$$

$$\underline{\mathbf{g}} \leq \mathbf{g} \leq \overline{\mathbf{g}} \tag{4}$$

$$\mathbf{A}\mathbf{f}_s + \mathbf{B}\mathbf{g}_s = \mathbf{d} \qquad \forall s \in \mathcal{S} \tag{5}$$

$$\mathbf{f}_s = \mathbf{S}\boldsymbol{\theta}_s \qquad \forall s \in \mathcal{S} \tag{6}$$

$$-\overline{\mathbf{f}} \leq \mathbf{f}_s \leq \overline{\mathbf{f}} \qquad \forall s \in \mathcal{S} \tag{7}$$

$$\underline{\mathbf{g}} \leq \mathbf{g}_s \leq \overline{\mathbf{g}} \qquad \forall s \in \mathcal{S}. \tag{8}$$

Equations (1)–(4) model the DC power flow in pre-contingency state and capture the nodal power balance (1), Kirchhoff's second law (2), transmission line limits (3), and generator limits (4). Analogously, equations (5)–(8) model the power flow for each

post-contingency state s. The bounds $\overline{\mathbf{g}}$ in (4) and (8) may be different from capacity $\hat{\mathbf{g}}$ due to commitment and/or operational constraints.

B. Automatic Primary Response

The APR is modeled as in [6], [7], [11]: under contingent state s, a global variable n_s is used to mimic the level of system response required for adjusting the power imbalance. The APR of generator i under contingency s, $g_{s,i} - g_i$, is proportional to its capacity $\hat{g}_i$ and to the parameter γ_i associated with the droop coefficient, i.e.,

$$g_{s,i} = \min\{g_i + n_s \gamma_i \hat{g}_i,\ \overline{g}_i\} \quad \forall i \in \mathcal{G}, \forall s \in \mathcal{S}, i \neq s \tag{9}$$

$$g_{s,s} = 0 \qquad \forall s \in \mathcal{S}. \tag{10}$$

The condition $i \neq s$ in (9) and the rest of this paper indicates that a constraint does not apply to generator s under the contingency s. i.e., the contingent state where this generator is offline.

Equations (9)–(10) are nonconvex and can be linearized by introducing binary variables $x_{s,i}$ to denote whether generator i in contingency s is not at its limit, i.e.,

$$|g_{s,i} - g_i - n_s \gamma_i \hat{g}_i| \leq \overline{g}_i (1 - x_{s,i}) \quad \forall i \in \mathcal{G}, s \in \mathcal{S}, i \neq s \tag{11}$$

$$g_i + n_s \gamma_i\, \hat{g}_i \geq \overline{g}_i (1 - x_{s,i}) \qquad \forall i \in \mathcal{G}, s \in \mathcal{S}, i \neq s \tag{12}$$

$$g_{s,i} \geq \overline{g}_i (1 - x_{s,i}) \qquad \forall i \in \mathcal{G}, s \in \mathcal{S}, i \neq s \tag{13}$$

$$n_s \in [0, 1] \qquad \forall s \in \mathcal{S} \tag{14}$$

$$x_{s,i} \in \{0, 1\} \qquad \forall i \in \mathcal{G}, s \in \mathcal{S} \tag{15}$$

$$g_{s,s} = 0 \qquad \forall s \in \mathcal{S}. \tag{16}$$

C. Extensive Formulation for the SCOPF Problem

The extensive formulation for the SCOPF problem using variables for generation, flows, and phase angles is as follows:

$$\min_{\boldsymbol{\theta}, \mathbf{f}, \mathbf{g}, [\boldsymbol{\theta}_s, \mathbf{f}_s, \mathbf{g}_s, n_s, \mathbf{x}_s]_{s \in \mathcal{S}}} \quad h(\mathbf{g}) \tag{17}$$

$$\text{s.t.:} \quad (1) - (4) \tag{18}$$

$$(5) - (16) \qquad \forall s \in \mathcal{S}. \tag{19}$$

Using power transfer distribution factors (PTDF), constraints (1)–(8) can be replaced by the following constraints:

$$\mathbf{e}^\top \mathbf{g} = \mathbf{e}^\top \mathbf{d} \tag{20}$$

$$|\mathbf{K}_0 (\mathbf{d} - \mathbf{B}\mathbf{g})| \leq \overline{\mathbf{f}} \tag{21}$$

$$(4) \tag{22}$$

$$\mathbf{e}^\top \mathbf{g}_s = \mathbf{e}^\top \mathbf{d} \qquad \forall s \in \mathcal{S} \tag{23}$$

$$|\mathbf{K}_0 (\mathbf{d} - \mathbf{B}\mathbf{g}_s)| \leq \overline{\mathbf{f}} \qquad \forall s \in \mathcal{S} \tag{24}$$

$$(8) \qquad \forall s \in \mathcal{S}. \tag{25}$$

Constraints (20)–(25) that involve the PTDF matrix $\mathbf{K}_0$ are from [36]. The total demand balance for the nominal and contingent states are enforced by (20) and (23) respectively. In constraints (21) and (24), the PTDF matrix translates the power injected by each generator at its bus into its contribution to the flow of each line. These constraints also bound the flows from above and below. Observe that $\mathbf{g}$ and $\mathbf{g}_s$ are the only variables in this formulation.

For conciseness, denote the power flow constraints (20)–(22) and (23)–(25) by $\mathbf{g} \in \mathcal{E}$ and $\mathbf{g}_s \in \mathcal{E}_s$ respectively. Similarly, denote the APR constraints (11)–(16) by $\mathcal{Y}_s = [\mathbf{g}, \mathbf{g}_s, \mathbf{x}_s, n_s] \in \mathcal{F}_s$. The extensive SCOPF formulation then becomes

$$\min_{\mathbf{g}, [\mathbf{g}_s, \mathbf{x}_s, n_s]_{s \in \mathcal{S}}} \quad h(\mathbf{g}) \tag{26}$$

$$\text{s.t.:} \quad \mathbf{g} \in \mathcal{E} \tag{27}$$

$$\mathbf{g}_s \in \mathcal{E}_s \qquad \forall s \in \mathcal{S} \tag{28}$$

$$\mathcal{Y}_s \in \mathcal{F}_s \qquad \forall s \in \mathcal{S}. \tag{29}$$

Note that the number of binary variables above grows quadratically with the number of generators. Hence, solving (26)–(29) becomes impractical for large-scale systems.

III. SCOPF Properties and CCGA

This section introduces key properties of the SCOPF problem and summarizes the CCGA proposed in [11]. These properties are necessary for the CCGA and the ML models. The CCGA is a decomposition scheme that serves both as a benchmark for evaluation and is used as part of the feasibility recovery scheme proposed in Section V.

Property 1: For $s \in \mathcal{S}$, given values $\mathbf{g}^*$ and n_s^* for $\mathbf{g}$ and n_s, there exists a unique value $\mathbf{g}_s^*$ for $\mathbf{g}_s$ that can be computed directly using constraints (11)–(16).

Property 2: Consider $s \in \mathcal{S}$ and a value $\mathbf{g}^*$ for $\mathbf{g}$. If there exists a value n_s^* for n_s that admits a feasible solution to constraints (11)–(16) and (23), then this value n_s^* is unique and can be computed by a simple bisection method [11].

Property 2 holds since, for a given $\mathbf{g}^*$, each component of $\mathbf{g}_s$ is continuous and monotone with respect to n_s. Hence the value n_s^* and its associated vector $\mathbf{g}_s^*$ that satisfy constraint (23) can be found by a simple bisection search over n_s.

Property 3: Constraint (24) can be formulated as

$$\mathbf{K}_1 \mathbf{g}_s + \mathbf{k}_2 \geq \mathbf{0} \tag{30}$$

$$\mathbf{K}_1 \mathbf{g}_s + \mathbf{k}_3 \geq \mathbf{0}, \tag{31}$$

using matrix operations to obtain $\mathbf{K}_1$, $\mathbf{k}_2$, and $\mathbf{k}_3$.

Note that each row of $\mathbf{K}_1$ and each element of $\mathbf{k}_2$ and $\mathbf{k}_3$ are associated with a specific transmission line. Therefore, for each $s \in \mathcal{S}$ and for each line, the (positive and negative) violation of the thermal limit of the line can be obtained by inspecting (30)–(31) for the proposed value $\mathbf{g}_s^{(*)}$.

A. The Column and Constraint Generation Algorithm

The CCGA, which relies on the above properties, alternates between solving a master problem to obtain a nominal schedule

$\mathbf{g}$ and a bisection method to obtain the state variables of each contingency. The master problem is specified as follows:

$$\min_{\mathbf{g},[\mathbf{g}'_s]_{s\in\mathcal{S}},[\mathbf{x}_s,n_s]_{s\in\mathbb{S}}} \quad h(\mathbf{g}) \tag{32}$$

$$\text{s.t.:} \quad \mathbf{g} \in \mathcal{E} \tag{33}$$

$$\mathbf{g}'_s - \mathbf{g} \le \overline{\mathbf{r}} \qquad \forall s \in \mathcal{S} \tag{34}$$

$$(8), (23), (16) \qquad \forall s \in \mathcal{S} \tag{35}$$

$$\mathcal{Y}_s \in \mathcal{F}_s \qquad \forall s \in \mathbb{S} \tag{36}$$

$$\mathbf{K}_1^l \mathbf{g}'_s + \mathbf{k}_2^l \ge \mathbf{0} \qquad \forall (l,s) \in \mathbb{U}^+ \tag{37}$$

$$\mathbf{K}_1^l \mathbf{g}'_s + \mathbf{k}_3^l \ge \mathbf{0} \qquad \forall (l,s) \in \mathbb{U}^-. \tag{38}$$

Constraints (32)–(38) uses variables $\mathbf{g}'_s$ to denote a "guess" for the post-contingency generation: the actual vector $\mathbf{g}_s$ is not determined by the master problem but by the aforementioned bisection method. Constraint (33) enforces the nominal state constraints. Constraint (34) imposes a valid bound for post-contingency generation. For all contingencies, constraint (35) enforces the generation capacity (8), total demand satisfaction (23), and the absence of generation for a failed generator (16). The APR is enforced "on-demand" in (36) for a reduced set of contingent states $\mathbb{S}$. Initially, $\mathbb{S} = \emptyset$. Inequalities (37)–(38) are also the "on-demand" versions of (30)–(31) for (a few) pairs of transmission lines and contingencies. Initially, $\mathbb{U}^+$ and $\mathbb{U}^-$ are empty sets.

The CCGA algorithm is specified in Algorithm 1. At iteration j, the master problem (32)–(38) computes $\mathbf{g}^j$. The bisection method then determines the contingent state variables $[\mathbf{g}_s^j, \mathbf{x}_s^j, n_s^j]_{s\in\mathcal{S}}$. The vectors $\boldsymbol{\tau}_s^+$ and $\boldsymbol{\tau}_s^-$ of positive numbers represent the positive and negative violations of transmission lines for contingent state s: they are calculated for all $s \in \mathcal{S}$ by inspecting constraints (30)–(31) for $[\mathbf{g}_s^j]_{s\in\mathcal{S}}$. The algorithm then computes the highest single line violation ϕ among all contingent states and uses s_ϕ to denote the contingent state associated with ϕ. The pairs lines/contingencies featuring violations above a predefined threshold β are added to the master problem by updating sets $\mathbb{U}^+$ and $\mathbb{U}^-$. Likewise, s_ϕ is added to $\mathbb{S}$. As a result, the variables and APR constraints associated with s_ϕ are added to the master problem during the next iteration. The CCGA terminates when $\phi < \epsilon$, where ϵ is the tolerance for line violation. The master problem is a MILP due to constraints (36) and the number of binary variables (at each iteration) is quadratic in the size of $\mathbb{S}$.

The following result from [11] ensures the correctness of CCGA: It shows that a solution to the master problem produces a nominal generation for which there exists a solution to each contingency that satisfies the APR and total demand constraints. Since the CCGA adds at least one violated line constraint and, possibly, a set of violated APR constraints for one contingency to the master problem at each iteration, it is guaranteed to converge after a finite number of iterations.

Theorem 1: For each solution $\mathbf{g}^*$ to the master problem, there exist values n_s^* and $\mathbf{g}_s^*$ that satisfy the demand constraint $\mathbf{e}^\top \mathbf{g}_s^* = \mathbf{e}^\top \mathbf{d}$ and the APR constraints (11)–(16) for each contingency s.

Proof: By (8) and (34), $g'_{s,i} \le \min\{\overline{g}_i, g_i + \gamma_i \hat{g}_i\}$ for each i and s, where $g'_{s,i}$ is the i-th element of $\mathbf{g}'_s$. When $n_s = 0, \mathbf{g}_s = \mathbf{g}$, except for $g_{s,s} = 0$. When $n_s = 1, g_{s,i} = \min\{\overline{g}_i, g_i + \gamma_i \hat{g}_i\} \ge g'_{s,i}$ for each i and s, with $i \ne s$. Since, by (23), $\mathbf{g}'_s$ meets the global demand, then $\mathbf{e}^\top \mathbf{g}_s \ge \mathbf{e}^\top \mathbf{g}'_s = \mathbf{e}^\top \mathbf{d}$ when $n_s = 1$. By the monotonicity and continuity of $g_{s,i}$ with respect to n_s (for a given g_i), there is a value n_s^* (associated with $\mathbf{g}_s^*$) that satisfies the demand constraint in (23) and preserves the APR constraints. □

Algorithm 1: CCGA.

```
Initialization: j ← 0, 𝕊 ← ∅, 𝕌⁺ ← ∅, 𝕌⁻ ← ∅
for j = 0, 1, ... do
    solve (32)–(36) to obtain g^j
    n_s^j ← apply the bisection method on all s ∈ 𝒮
    g_s^j ← enforce (11)–(16) on all s ∈ 𝒮
    τ_s^-, τ_s^+ ← get the line violations of g_s^j using
    (30)–(31) for all s ∈ 𝒮
    φ ← compute the highest line violation among all
    s ∈ 𝒮
    s_φ ← select the contingent state associated with φ
    𝕊 ← 𝕊 ∪ {s_φ}
    𝕌⁺ ← 𝕌⁺ ∪ { (l, s) | τ_s^+[l] > β}
    𝕌⁻ ← 𝕌⁻ ∪ { (l, s) | τ_s^-[l] > β}
    BREAK if φ ≤ ε.
end for
```

IV. Deep Neural Networks for SCOPF

This section describes the use of supervised learning to obtain DNNs that map a load vector into a solution of the SCOPF problem. A DNN consists of many layers, where the input for each layer is typically the output of the previous layer [37]. This work uses fully-connected DNNs.

A. Specification of the Learning Problem

For didactic purposes, the specification of the learning problem uses the extensive formulation (26)–(29). The training data is a collection of instances of the form

$$\{\mathbf{d}^t; \mathbf{g}^t, [\mathbf{g}_s^t, n_s^t, \mathbf{x}_s^t]_{s\in\mathcal{S}}\}_{t\in\mathcal{T}}$$

where $(\mathbf{g}^t, [\mathbf{g}_s^t, n_s^t, \mathbf{x}_s^t]_{s\in\mathcal{S}})$ is the optimal solution (ground truth) to the SCOPF problem for input $\mathbf{d}^t$. The DNN is a parametric function $\mathrm{O}[\boldsymbol{\omega}](\cdot)$ whose parameters are the network $\boldsymbol{\omega}$: It maps a load vector $\mathbf{d}$ into an approximation $\mathrm{O}[\boldsymbol{\omega}](\mathbf{d}) = \{\dot{\mathbf{g}}, [\dot{n}_s, \dot{\mathbf{x}}_s, \dot{\mathbf{g}}_s]_{s\in\mathcal{S}}\}$ of the optimal solution to the SCOPF problem for load $\mathbf{d}$. The goal of the machine-learning training to find the optimal weights $\boldsymbol{\omega}^*$, i.e.,

$$\boldsymbol{\omega}^* = \arg\min_{\boldsymbol{\omega}} \sum_{t\in\mathcal{T}} \mathbb{L}_0^t(\dot{\mathbf{g}}^t) + \sum_{s\in\mathcal{S}} \mathbb{L}_s^t(\dot{\mathbf{g}}_s^t, \dot{\mathbf{x}}_s^t, \dot{n}_s^t) \tag{39}$$

$$\text{s.t.: } O[\boldsymbol{\omega}](\mathbf{d}^t) = (\dot{\mathbf{g}}^t, [\dot{\mathbf{g}}_s^t, \dot{n}_s^t, \dot{\mathbf{x}}_s^t]) \qquad \forall t \in \mathcal{T} \tag{40}$$

$$\dot{\mathbf{g}}^t \in \mathcal{E}^t \qquad \forall t \in \mathcal{T} \tag{41}$$

$$\dot{\mathbf{g}}_s^t \in \mathcal{E}_s^t \qquad \forall t \in \mathcal{T}, \forall s \in \mathcal{S} \tag{42}$$

$$\dot{\mathcal{Y}}_s^t \in \mathcal{F}_s^t \qquad \forall t \in \mathcal{T}, \forall s \in \mathcal{S} \tag{43}$$

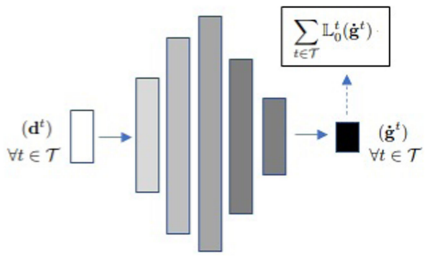

Fig. 1. Baseline Model: Each layer is fully connected with softplus activation. White box on the left corresponds to input tensor (demand), dark, colored, boxes correspond to output layers. Loss component is shown as a white rectangle connected with the output layer by a dashed arrow.

where the loss functions are defined as

$$\mathbb{L}_0^t(\dot{\mathbf{g}}^t) = ||\mathbf{g}^t - \dot{\mathbf{g}}^t||_2$$
$$\mathbb{L}_s^t(\dot{\mathbf{g}}_s^t, \dot{\mathbf{x}}_s^t, \dot{n}_s^t) = ||\mathbf{g}_s^t - \dot{\mathbf{g}}_s^t||_2 + ||\mathbf{x}_s^t - \dot{\mathbf{x}}_s^t||_2 + ||n_s^t - \dot{n}_s^t||_2$$

and minimize the distance between the prediction and the ground truth. There are two difficulties in this learning problem: the large number of scenarios, variables, and constraints, and the satisfaction of constraints (41)–(43). This section examines possible approaches.

B. The Baseline Model

The baseline model is a parsimonious ML approach that disregards constraints (41)–(43) and predicts the nominal generation only, i.e.,

$$\boldsymbol{\omega}^* = \quad \arg\min_{\boldsymbol{\omega}} \sum_{t\in\mathcal{T}} \mathbb{L}_0^t(\dot{\mathbf{g}}^t)$$
$$\text{s.t.: } O[\boldsymbol{\omega}](\mathbf{d}^t) = (\dot{\mathbf{g}}^t) \qquad \forall t \in \mathcal{T}.$$

It uses a DDN model with linear layers interspersed with nonlinear layers that use the softplus activation function. The sizes of the input and output of each layer are linearly proportional to $|\mathcal{N}|$ and $|\mathcal{G}|$. A high-level algebraic description of layers of the DNN follows:

$$\mathbf{l}_i = \pi(\mathbf{W}_i\mathbf{l}_{i-1} + \mathbf{b}_i), \qquad \text{for each layer } \mathbf{l}_i$$
$$\mathbf{l}_1 = \pi(\mathbf{W}_1\mathbf{d} + \mathbf{b}_1).$$

The elements of vector $\boldsymbol{\omega}$ are rearranged as matrices $\mathbf{W}_i$ and vector of biases $\mathbf{b}_i$. Note that the demand vector $\mathbf{d}$ is the input for the first layer. The symbol π denotes a nonlinear activation function. The baseline model is represented in Fig. 1.

Unfortunately, training the baseline model tends to produce predictors violating the problem constraints [34], [35].

C. A Lagrangian Dual Model for Nominal Constraints

The DNN model of this section extends the baseline model. While not directly used in the results, it constitutes a relevant didactic step in the construction of the final ML model of Section IV-D.

Algorithm 2: Lagrangian Dual Model ($\mathcal{T}$, $\mathbb{C}$, α, ρ, $\boldsymbol{\lambda}^0$, $\boldsymbol{\omega}^0$).

1: $j \leftarrow 0$.
2: **for** $j = 0, 1, \ldots$ **do**
3: **for** $k = 0, 1, \ldots$ **do**
4: Sample minibatch: $\mathbb{T}^k \subset \mathcal{T}$
5: **for** $t \in \mathbb{T}^k$ **do**
6: Compute $\mathrm{O}[\boldsymbol{\omega}^j](\mathbf{d}^t) = \dot{\mathbf{g}}^t$ and $\mathbb{L}_0^t(\dot{\mathbf{g}}^t)$
7: Compute $\nu_c^t(\dot{\mathbf{g}}^t)\ \forall c \in \mathbb{C}$
8: **end for**
9: $\boldsymbol{\omega}^j \leftarrow \boldsymbol{\omega}^j - \alpha$
$\nabla_{\boldsymbol{\omega}^j}[\sum_{t\in\mathcal{T}}(\mathbb{L}_0^t(\dot{\mathbf{g}}^t) + \sum_{c\in\mathbb{C}} \lambda_c\nu_c^t(\dot{\mathbf{g}}^t))]$
10: **end for**
11: $\lambda_c^{j+1} \leftarrow \lambda_c^j + \rho\,\tilde{\nu}_c\ \forall c \in$ (41)
12: $\boldsymbol{\omega}^{j+1} \leftarrow \boldsymbol{\omega}^j$
13: **end for**

This section expands the baseline model to include constraints on the nominal state (41). Constraints (42)–(43) on the contingency cases are not considered in the model. To capture physical and operational constraints, the training of the DNN adopts the Lagrangian dual approach from [35].

The Lagrangian dual approach relies on the concept of constraint violations. The violations of a constraint $f(x) = 0$ is given by $|f(x)|$, while the violations of $f(x) \geq 0$ are specified by $\max(0, -f(x))$. Although these expressions are not differentiable, they admit subgradients. Let $\mathbb{C}$ represent the set of nominal constraints and $\nu_c(\mathbf{g})$ be the violations of constraint c for generation dispatch $\mathbf{g}$. The Lagrangian dual approach introduces a term $\lambda_c\nu_c^t(\mathbf{g}^t)$ in the objective function for each $c \in \mathbb{C}$ and each $t \in \mathcal{T}$, where λ_c is a Lagrangian multiplier. The optimization problem then becomes

$$LR(\boldsymbol{\lambda}) = \quad \min_{\boldsymbol{\omega}} \quad \sum_{t\in\mathcal{T}}(\mathbb{L}_0^t(\dot{\mathbf{g}}^t) + \sum_{c\in\mathbb{C}} \lambda_c\nu_c^t(\dot{\mathbf{g}}^t)) \qquad (44)$$
$$\text{s.t.} \quad O[\boldsymbol{\omega}](\mathbf{d}^t) = (\dot{\mathbf{g}}^t) \qquad \forall t \in \mathcal{T} \qquad (45)$$

and the Lagrangian dual is simply

$$LD = \max_{\boldsymbol{\lambda}} LR(\boldsymbol{\lambda}). \qquad (46)$$

Problem (46) is solved by iterating between training for weights $\boldsymbol{\omega}$ and updating the Lagrangian multipliers. Iteration j uses Lagrangian multiplier $\boldsymbol{\lambda}^j$ and solves $LR(\boldsymbol{\lambda}^j)$ to obtain the optimal weights $\boldsymbol{\omega}^j$. It then updates the Lagrangian multipliers using the constraint violations. The overall scheme is presented in Algorithm 2. Lines 3–10 train weights $\boldsymbol{\omega}^j$ for a fixed vector of Lagrangian multipliers $\boldsymbol{\lambda}^j$, using minibatches and a stochastic gradient descent method with learning rate α. For each minibatch, the algorithm computes the predictions (line 6), the constraint violations (line 7), and updates the weights (line 9). Lines 2–13 describe the solving of Lagrangian dual. It computes the Lagrangian relaxation described previously and updates the Lagrangian multipliers in line 11 using the median violation $\tilde{\nu}_c$ for each nominal constraint c.

D. CCGA-DNN Model

This section presents the final ML model, the CCGA-DNN, which mimics a CCGA algorithm. In particular, the CCGA-DNN combines the Lagrangian dual model of Section IV-C with an outer loop that adds constraints for the contingent states on-demand. Observe first that a direct Lagrangian dual approach to the SCOPF would require an outer loop to add predictors $[\dot{\mathbf{g}}_s, \dot{\mathbf{x}}_s, \dot{n}_s]_{s\in\mathcal{S}}$ and constraints (42)–(43) for selected contingency states s. Unfortunately, the addition of new predictors structurally modifies the DNN output $\mathrm{O}[\boldsymbol{\omega}](\cdot)$ and induces a considerable increase in the DNN size.

The key ideas to overcome this difficulty are the following:

1) The CCGA-DNN mimics the CCGA and replaces the output of the master problem by the predictions $O[\boldsymbol{\omega}^j](\mathbf{d}^t), \forall t$ at iteration j;
2) As is typical in robust optimization scheme, the CCGA-DNN only selects a small number of contingent states to be penalized in the DNN (master problem) at each iteration;
3) The CCGA-DNN replaces constraints (43) by constraints of the form

$$\dot{g}^t_{s,i} = \max\{0, \min\{\dot{g}^t_i + \dot{n}^t_s \gamma_i \, \hat{g}_i \, , \overline{g}_i\}\}, \qquad (47)$$

where constraints (47) are not differentiable but admit subgradients and hence can be dualized in the objective function. In this setting, $\dot{n}^t_s$ is obtained by the bisection method on the prediction $\dot{g}^t$ obtained in previous iteration.

It is interesting to highlight that point (3) allows for penalizing contingent state constraints while avoiding adding predictors for these states in the DNN.

The CCGA-DNN is summarized in Algorithm 3 and depicted in Fig. 2. It is initialized in line 1 with the set of constraints $\mathbb{C}$ encompassing the nominal constraints (41) only and Lagrangian multipliers set to zero. Thus, the first iteration of the main loop (lines 3–28) returns DNN weights equivalent to those of the baseline model. At each iteration j, the training loop (lines 5–12) produces updated weights $\boldsymbol{\omega}^j$. Next, a post-training loop (lines 13–20) applies the bisection method to find $\dot{n}^t_s$ for all t and APR constraints (11)–(16) to obtain $\dot{\mathbf{g}}^t_s$ (lines 15–16). These values are then used to compute the nominal violations (line 17), and the highest line violation ϕ^t among all states (line 18), as well as its associated contingent state s^t_ϕ. The post-training loop also increases the element of the counter vector $\mathbf{p}$ associated with s^t_ϕ whenever the highest violation ϕ^t at iteration t is above tolerance ϵ (line 19). Then, in the main loop, contingency states with high frequencies of violated lines $\mathbb{S}'$ are identified (line 21) using a threshold β_1. The algorithm terminates when $\mathbb{S}'$ is empty and the median relative violations for the nominal constraints in (41) are within the tolerances β_c (line 22). Otherwise, the set of constraints $\mathbb{C}$ is updated with constraints (30)–(31) and (47) for the added contingent states $\mathbb{S}' \setminus \mathbb{S}$ (line 23). The Lagrangian multipliers for nominal constraints are updated in line 24. The multipliers for the additional constraints (30)–(31) ($s \in \mathbb{S}'$) are initialized in line 25. Finally, the multipliers for constraints (30)–(31) ($s \in \mathbb{S}$) are all updated with $\tilde{\phi}$ (line 27), where $\tilde{\phi}$ is the median ϕ^t among all ϕ^t for $t \in \mathcal{T}$. Note that the process of updating Lagrangian multipliers for (30)–(31) ($s \in \mathbb{S}$) is different and much stricter than that for nominal constraints in line 24.

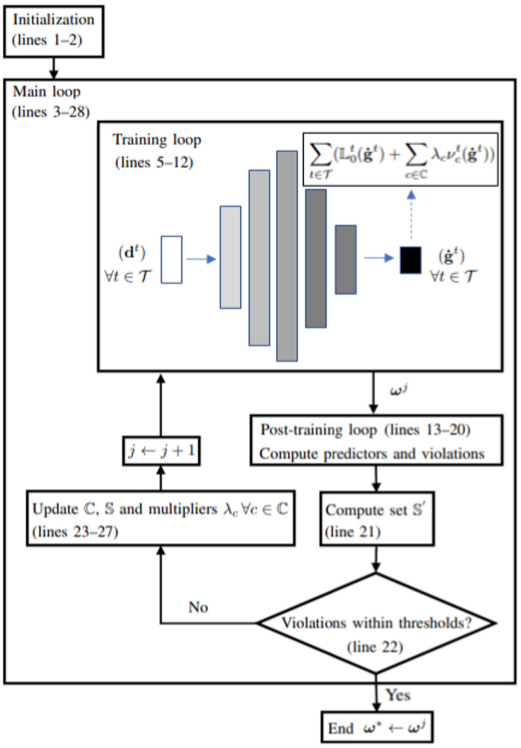

Fig. 2. Flowchart for CCGA-DNN (Algorithm 3).

TABLE I
INSTANCE SIZE FOR THE SCOPF PROBLEM (26)–(29) AFTER PRESOLVE

System	$\lvert\mathcal{G}\rvert$	$\lvert\mathcal{L}\rvert$	$\lvert\mathcal{N}\rvert$	Total Variables	Binary Variables	Linear Constraints
118-IEEE	54	186	118	13,466	2,862	19,137
1354-PEG	260	1,991	1,354	387,026	63,455	513,677
1888-RTE	297	2,531	1,888	467,011	79,032	624,780

V. FEASIBILITY RECOVERY

The training step produces a set of weights $\boldsymbol{\omega}^*$ and the associated DNN produces, almost instantly, a dispatch prediction $\dot{\mathbf{g}} = O[\boldsymbol{\omega}^*](\mathbf{d})$ for an input load vector $\mathbf{d}$. However, the prediction $\dot{\mathbf{g}}$ may violate the nominal and contingency constraints. To restore feasibility, this paper proposes a feasibility-recovery CCGA, denoted by FR-CCGA, that finds the feasible solution closest to $\dot{\mathbf{g}}$. The FR-CCGA scheme also follows Algorithm 1, except for a modification in the master problem. The master problem for FR-CCGA is similar to (32)–(38) but it uses a different objective function, i.e.,

$$\min_{\mathbf{g},[\mathbf{g}'_s]_{s\in\mathcal{S}},[\mathbf{x}_s,n_s]_{s\in\mathbb{S}}} \quad \|\dot{\mathbf{g}} - \mathbf{g}\| \qquad (48)$$

$$\text{s.t.:} \quad (33)\text{–}(38) \qquad \forall\, \mathcal{S},\, \mathbb{S},\, \mathbb{U}^+,\, \mathbb{U}^-. \qquad (49)$$

Note that $\dot{\mathbf{g}}$ is a constant vector in FR-CCGA.

While CCGA and FR-CCGA are similar decomposition schemes in nature, featuring the same rationale for adding constraints and variables (linear and binaries) to respective master

Algorithm 3: CCGA-DNN ($\mathcal{T}$, α, ρ, β_1, β_c, ϵ).

1: $\mathbb{C} \leftarrow \{(41)\}$, $\mathbb{S} \leftarrow \emptyset$, $\boldsymbol{\lambda}^0 \leftarrow \mathbf{0}$, $\boldsymbol{\omega}^0 \leftarrow \mathbf{0}$
2: Create a counter vector $\mathbf{p}$ of size $|\mathcal{S}|$
3: **for** $j = 0, 1, \ldots$ **do**
4: $\quad \mathbf{p} \leftarrow \mathbf{0}$
5: $\quad$ **for** $k = 0, 1, \ldots$ **do**
6: $\quad\quad$ Sample minibatch: $\mathbb{T}_k \subset \mathcal{T}$
7: $\quad\quad$ **for** $t \in \mathbb{T}_k$ **do**
8: $\quad\quad\quad$ Compute $\mathrm{O}[\boldsymbol{\omega}^j](\mathbf{d}^t) = \dot{\mathbf{g}}^t$ and $\mathbb{L}_0^t(\dot{\mathbf{g}}^t)$
9: $\quad\quad\quad$ Compute $\nu_c^t(\dot{\mathbf{g}}^t), \forall c \in \mathbb{C}$
10: $\quad\quad$ **end for**
11: $\quad\quad$ $\boldsymbol{\omega}^j \leftarrow \boldsymbol{\omega}^j - \alpha \nabla_{\boldsymbol{\omega}^j} [\sum_{t \in \mathcal{T}} (\mathbb{L}_0^t(\dot{\mathbf{g}}^t) + \sum_{c \in \mathbb{C}} \lambda_c \nu_c^t(\dot{\mathbf{g}}^t))]$
12: $\quad$ **end for**
13: $\quad$ **for** $t \in \mathcal{T}$ **do**
14: $\quad\quad$ $\dot{\mathbf{g}}^t \leftarrow \mathrm{O}[\boldsymbol{\omega}^j](\mathbf{d}^t)$
15: $\quad\quad$ $\dot{n}_s^t \leftarrow$ bisection method, $\forall s \in \mathcal{S}$
16: $\quad\quad$ $\dot{\mathbf{g}}_s^t \leftarrow$ enforce (11)–(16), $\forall s \in \mathcal{S}$
17: $\quad\quad$ Compute $\nu_c^t(\dot{\mathbf{g}}^t), \forall c \in (41)$
18: $\quad\quad$ Compute ϕ^t and identify s_ϕ^t
19: $\quad\quad$ **if** $\phi^t > \epsilon$ **then** increase (s_ϕ^t)-th element of $\mathbf{p}$ by 1
20: $\quad$ **end for**
21: $\quad$ $\mathbb{S}' \leftarrow \{s \,|\, \mathbf{p}[s] \,/\, |\mathcal{T}| > \beta_1\}$
22: $\quad$ **BREAK** if $\mathbb{S}' \equiv \emptyset$ and $\tilde{\nu_c} \leq \beta_c, \forall c \in (41)$.
23: $\quad$ $\mathbb{C} \leftarrow \mathbb{C} \cup \{(30) - (31), (47), \forall s \in (\mathbb{S}' \setminus \mathbb{S})\}$
24: $\quad$ $\lambda_c^{j+1} \leftarrow \lambda_c^j + \rho\, \tilde{\nu}_c \; \forall c \in (41)$
25: $\quad$ $\lambda_{(30)}^{j+1}, \lambda_{(31)}^{j+1} \leftarrow 0, \; \forall s \in (\mathbb{S}' \setminus \mathbb{S})$
26: $\quad$ $\mathbb{S} \leftarrow \mathbb{S} \cup \mathbb{S}'$
27: $\quad$ $\lambda_{(30)}^{j+1}, \lambda_{(31)}^{j+1} \leftarrow \lambda_{(30)}^{j}, \lambda_{(31)}^{j} + \rho\, \tilde{\phi}, \; \forall s \in \mathbb{S}$
28: **end for**

problems, FR-CCGA is significantly faster because $O[\boldsymbol{\omega}^*](\mathbf{d})$ is often close to feasibility, which results in fewer iterations.

VI. Computational Experiments

A. Data

The test cases are based on modified versions of 3 traditional benchmark system topologies from [38], namely, the 118-IEEE, the 1354-PEG (PEGASE) system, and the 1888-RTE system. Table I reports the size of each system topology (numbers of generators, lines, and buses) and the number of constraints and variables for a single instance of the SCOPF problem for each benchmark. As can be seen, the larger system topologies encompass thousands of lines and buses, resulting in an extensive formulation for the SCOPF problem (26)–(29) (after Gurobi's presolve) with hundreds of thousands of constraints and variables, and with dozens of thousands of binary variables.

For each topology, training and testing data are given by the inputs and solutions of many instances that are constructed as follows. For each instance, the net demand of each bus has a deterministic component and a random component. The deterministic component varies across instances from 82% of the nominal net load to near-infeasibility values by small increments of 0.002%. The random component for each bus of each instance is independently and uniformly distributed ranging from -0.5% to 0.5% of the corresponding nominal nodal net load. For each system topology, thousands of instances were generated for training and testing, namely, 14 000 instances for 118-IEEE, 10 000 instances for 1354-PEG, and 14 000 instances for 1888-RTE. For each topology, the training set $\mathcal{T}$ is composed by 70% of the instances, selected randomly.

TABLE II
Network Architecture

Layer	Type	Size in	Size out	Activation Function
Initial	Fully connected	$\lvert\mathcal{N}\rvert$	$2\,\lvert\mathcal{N}\rvert + 2\,\lvert\mathcal{G}\rvert$	Softplus
2	Fully connected	$2\,\lvert\mathcal{N}\rvert + 2\,\lvert\mathcal{G}\rvert$	$4\,\lvert\mathcal{N}\rvert + 4\,\lvert\mathcal{G}\rvert$	Softplus
3	Fully connected	$4\,\lvert\mathcal{N}\rvert + 4\,\lvert\mathcal{G}\rvert$	$8\,\lvert\mathcal{N}\rvert + 8\,\lvert\mathcal{G}\rvert$	Softplus
4	Fully connected	$8\,\lvert\mathcal{N}\rvert + 8\,\lvert\mathcal{G}\rvert$	$16\,\lvert\mathcal{G}\rvert$	Softplus
Final	Fully connected	$16\,\lvert\mathcal{G}\rvert$	$\lvert\mathcal{G}\rvert$	Softplus

Algorithm 1 was applied to tackle these instances for a maximum line violation of $\epsilon = 0.05$ MW and an optimality gap of 0.25%. This algorithm was implemented in Julia 0.6.4 with the JuMP modeling package using Gurobi 8.1.1 as a backbone solver. The task of solving these instances was performed by multiple nodes of the PACE cluster at Georgia Institute of Technology. It is relevant to highlight that CPU times for this task were not stored or used for time comparisons in this work.

B. Network Architecture and Training Aspects

Both the baseline model and the CCGA-DNN model use the same network architecture, displayed in Table II, where all layers are fully-connected and the nonlinear softplus activation function is applied. The sizes of inputs and outputs of the DNN layers are parameterized in terms of number of buses and generators, thereby varying among system topologies. As can be seen, the size of the input for the initial layer ($|\mathcal{N}|$) is consistent with the cardinality of $\mathbf{d}$, while the size of output of the last layer ($|\mathcal{G}|$) matches the cardinality of $\mathbf{g}$.

Algorithm 3 was applied for training with ϵ set to 1 MW, β_1 to 5%, β_c to $1.5 \cdot 10^{-2}$, and ρ to 10^5. The inner training loop of Algorithm 3 (lines 5–12) is executed $1.5 \cdot 10^5$ times with a learning rate α varying from 10^{-4} to 10^{-10}. Note that the first iteration of Algorithm 3 returns the weights of the baseline model, while the final iteration return the weights of the CCGA-DNN.

The DNN models were implemented using PyTorch package with Python 3.0 and the training was performed using NVidia Tesla V100 GPUs and 2 GHz Intel Cores. The use of GPUs is paramount in this task since it allows for the faster computation of multiple parallel processes. In the following, for conciseness, the baseline model is denoted by $\mathcal{M}_b$ and the CCGA-DNN by $\mathcal{M}_{ccga}$.

Table III presents a training summary. As expected, the $\mathcal{M}_b$ requires lower training times (last column), below 12 hours, for all topologies. This is consistent with the higher number of iterations (column 3) of the $\mathcal{M}_{ccga}$, which incrementally adds

TABLE III
TRAINING SUMMARY

System	Model	Iterations	Added Contingencies at last iteration ($\mathbb{S}$)	Time (hours)
118-IEEE	$\mathcal{M}_b$	1	0	1.1
	$\mathcal{M}_{ccga}$	3	1	3.4
1354-PEG	$\mathcal{M}_b$	1	0	11.1
	$\mathcal{M}_{ccga}$	3	7	43.1
1888-RTE	$\mathcal{M}_b$	1	0	9.6
	$\mathcal{M}_{ccga}$	2	2	19.3

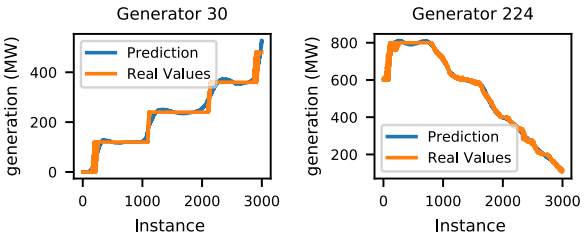

Fig. 3. Prediction of $\mathcal{M}_{ccga}$ for selected generators of the 1354-PEG System.

TABLE IV
PREDICTION MEAN ABSOLUTE ERRORS (%)

System	Model	Generation Range (MW) 10 50	50 100	100 250	250 500	500 1000	1000 2000	2000 5000
118-IEEE	$\mathcal{M}_b$	2.3	2.8	0.7	0.3	N/A	N/A	N/A
	$\mathcal{M}_{ccga}$	2.5	3.0	0.7	0.4	N/A	N/A	N/A
1354-PEG	$\mathcal{M}_b$	2.4	1.3	1.1	0.9	0.4	0.2	0.1
	$\mathcal{M}_{ccga}$	5.0	1.8	1.2	1.0	0.4	0.3	0.2
1888-RTE	$\mathcal{M}_b$	1.3	1.2	0.7	0.4	0.3	0.1	N/A
	$\mathcal{M}_{ccga}$	1.4	1.1	0.6	0.4	0.3	0.1	N/A

Lagrangian penalty terms for contingent states. Even though all contingencies are eligible to be added to the $\mathcal{M}_{ccga}$, the results (column 4) show that adding only a few contingent states (to set $\mathbb{S}$) is sufficient to "protect" against all contingent states. This result is typical in robust optimization applications.

C. Prediction Quality

Accurate predictions were obtained for all DNN models and topologies. Figure 3 illustrates how $\mathcal{M}_{ccga}$ can learn complex generator patterns arising in the 1354-PEG system. Table IV reports the mean absolute errors for predictions $\dot{\mathbf{g}}^t$, segmented by generation range. As expected, the predictions are relatively more accurate for larger generation labels. This happens because the loss function is parameterized in terms of total error and not relative error. It can also be observed that $\mathcal{M}_b$ achieves a slightly better overall accuracy which is expected since it is the less constrained model.

Table V reports selected indicators of violations: the relative violation $\nu_{(20)}$ of the total load constraint and the relative violation RLV of the lines associated with ϕ. The results report median values, as well as lower and upper bounds for intervals that capture 95% of the instances. Both models achieve the desired tolerance of $\beta_c = 1.5 \cdot 10^{-2}$ for $\nu_(20)$ (the tolerance β_c does not apply to RLV). Model $\mathcal{M}_{ccga}$ produces lower overall violations

TABLE V
SELECTED INDICATORS OF VIOLATION ACROSS INSTANCES (%)

System	Model	$\nu_{(20)}$ Median	$\nu_{(20)}$ 95%-Interval		RLV Median	RLV 95%-Interval	
118-IEEE	$\mathcal{M}_b$	0.016	0.001	0.055	0.099	0.000	0.467
	$\mathcal{M}_{ccga}$	0.003	0.000	0.011	0.000	0.000	0.548
1354-PEG	$\mathcal{M}_b$	0.018	0.001	0.062	0.259	0.086	1.256
	$\mathcal{M}_{ccga}$	0.010	0.000	0.040	0.005	0.000	0.117
1888-RTE	$\mathcal{M}_b$	0.012	0.001	0.047	0.205	0.021	1.418
	$\mathcal{M}_{ccga}$	0.005	0.000	0.027	0.007	0.000	0.057

$\nu_{(20)}$ – Net load constraint violation divided by total load.
RLV – Relative violation for line associated with ϕ.

TABLE VI
CPU TIME COMPARISON

System	Model	Median	Mean	Min.	Max.	Std.
118-IEEE	CCGA	0.21	0.21	0.10	3.72	0.31
	CCGA-H	0.18	0.20	0.14	3.54	0.24
	FR-$\mathcal{M}_b$	0.02	0.06	0.02	1.58	0.16
	FR-$\mathcal{M}_{ccga}$	0.03	0.07	0.02	1.37	0.17
1354-PEG	CCGA	321.75	327.21	75.59	741.80	127.10
	CCGA-H	5.26	5.32	4.40	8.65	0.49
	FR-$\mathcal{M}_b$	5.34	8.43	1.32	133.37	13.51
	FR-$\mathcal{M}_{ccga}$	1.52	2.17	0.77	8.45	1.74
1888-RTE	CCGA	5.48	7.41	3.11	30.92	7.07
	CCGA-H	3.17	3.15	2.31	13.55	0.91
	FR-$\mathcal{M}_b$	5.32	5.50	1.22	18.07	3.35
	FR-$\mathcal{M}_{ccga}$	2.12	1.95	0.91	4.54	0.92

and has a major effect on RLV. As an example, $\mathcal{M}_{ccga}$ predictions are quasi-feasible for 1354-PEG for all contingencies: For 95% of instances $t \in \mathcal{T}$, the relative violation of the single most violated line (among all lines and contingencies in instance t) is below 0.117%.

D. Comparison With Benchmarks

The previous sections reported on the accuracy of the predictors. This section shows how FR-CCGA leverages the predictors to find near-optimal feasible primal solutions. More precisely, experiments involving 200 randomly selected instances for each system topology compare in terms of cost, number of iterations, and CPU time, benchmark approaches with FR-CCGA which is seeded with $\mathcal{M}_b$ (labeled as FR-$\mathcal{M}_b$) or $\mathcal{M}_{ccga}$ (labeled as FR-$\mathcal{M}_{ccga}$). Two benchmark approaches are used for comparison: the CCGA (Algorithm 1) and the CCGA-H. The latter is the application of Algorithm 1 to an heuristic version of SCOPF, where the binary variables in $\mathbf{x}_s$ are set to 1. Thus, it assumes the APR is always in the linear phase of response (note that this is the typical outcome for most generators under most contingencies). The CCGA-H, therefore, is a much simpler version of the CCGA, where the master problem at each iteration is a linear program.

Each instance for each approach was solved with the same tolerances used for training and under the same hardware condition, on a laptop Dell XPS 13 9380 featuring a i7-8565 U processor at 1.8 GHz and 16 GB of RAM. Respective algorithms were implemented in Julia 0.6.4 with JuMP modeling package using Gurobi 8.1.1 as the primary solver. Tables VI, VII, and VIII summarize the experiments.

TABLE VII
ITERATIONS UNTIL CONVERGENCE

System	Model	Mean	Min.	Max.
118-IEEE	CCGA	2.525	2.000	3.000
	CCGA-H	2.525	2.000	3.000
	FR-$\mathcal{M}_b$	1.090	1.000	2.000
	FR-$\mathcal{M}_{ccga}$	1.135	1.000	3.000
1354-PEG	CCGA	7.780	7.000	12.000
	CCGA-H	7.680	6.000	10.000
	FR-$\mathcal{M}_b$	3.555	2.000	7.000
	FR-$\mathcal{M}_{ccga}$	1.900	1.000	5.000
1888-RTE	CCGA	2.530	2.000	4.000
	CCGA-H	2.505	2.000	3.000
	FR-$\mathcal{M}_b$	2.930	1.000	6.000
	FR-$\mathcal{M}_{ccga}$	1.710	1.000	3.000

TABLE VIII
COST INCREASE OVER CCGA (%)

System	Model	Median	Mean	Min.	Max.	Std.
118-IEEE	CCGA-H	0.001	0.001	0.000	0.010	0.002
	FR-$\mathcal{M}_b$	0.021	0.019	-0.073	0.055	0.014
	FR-$\mathcal{M}_{ccga}$	0.027	0.030	-0.010	0.112	0.020
1354-PEG	CCGA-H	0.883	0.880	0.849	0.890	0.010
	FR-$\mathcal{M}_b$	0.020	0.021	-0.007	0.051	0.012
	FR-$\mathcal{M}_{ccga}$	0.067	0.067	0.032	0.091	0.017
1888-RTE	CCGA-H	0.349	0.340	0.323	0.357	0.013
	FR-$\mathcal{M}_b$	0.026	0.024	-0.003	0.070	0.011
	FR-$\mathcal{M}_{ccga}$	0.033	0.033	0.004	0.070	0.013

Table VI reports statistics for computation times. The CCGA has the worst overall performance. The CCGA-H performs well in all systems, even though it is one order of magnitude slower than both FR-$\mathcal{M}_b$ and FR-$\mathcal{M}_{ccga}$ for the 118-IEEE. For the 1354-PEG system, the most challenging network by far, the FR-$\mathcal{M}_{ccga}$ performs significantly better than its counterparts. The FR-$\mathcal{M}_{ccga}$ is also more robust than the FR-$\mathcal{M}_b$. The excellent performance of the FR-$\mathcal{M}_{ccga}$ relates to the near-feasibility of its associated DNN prediction, as discussed in Section VI-C. This proximity to feasibility results in very few iterations of the feasibility recovery algorithm, which is reported in Table VII along with iterations for all approaches. As is clear, FR-$\mathcal{M}_{ccga}$ requires fewer iterations for larger systems.

Table VIII reports the cost/objective increase of FR-$\mathcal{M}_b$, FR-$\mathcal{M}_{ccga}$, and CCGA-H over CCGA, which is exact (modulo tolerances). With the exception of the smaller system, where all approaches perform well, CCGA-H is worse than both FR-$\mathcal{M}_b$ and FR-$\mathcal{M}_{ccga}$. The results for the complex 1354-PEG system show that the cost increase for the CCGA-H is more than one order of magnitude higher than those for FR-$\mathcal{M}_b$ and FR-$\mathcal{M}_{ccga}$. Overall these results show a very small cost increase for both FR-$\mathcal{M}_b$ and FR-$\mathcal{M}_{ccga}$ over CCGA, suggesting that the proposed approaches are promising, not only in terms of CPU times but also cost-wise.

Finally, Fig. 4 exemplifies that FR-CCGA and CCGA can be further combined to produce real-time optimality gaps. The illustration shows the behavior across time of FR-$\mathcal{M}_{ccga}$ and CCGA on a randomly chosen instance of the 1354-PEG system. The red line represents the upper bound (final feasible solution) generated in 1.87 seconds by the FR-$\mathcal{M}_{ccga}$. The blue line represents a sequence of true lower bounds (infeasible solutions) generated by the intermediary iterations of Algorithm 1 (for CCGA). This indicate that the FR-CCGA and CCGA can be run in parallel (in separate threads) to provide upper and lower bounds to the SCOPF respectively. This may be valuable for operators to assess the quality of the associated FR-CCGA solution and decide whether to commit to the already feasible FR-CCGA solutions or wait until a better solution is found or the optimality gap is sufficiently small.

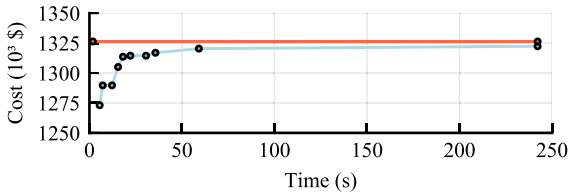

Fig. 4. Convergence plot for the 1354-PEG system.

VII. EXTENSIONS, LIMITATIONS AND FUTURE WORK

There are several interesting avenues for future research. One of them is the addition of new features to the SCOPF such as the inclusion of line contingencies. This requires the precomputation of different structures for $\mathbf{K}_0$, $\mathbf{K}_1$, $\mathbf{k}_2$, and $\mathbf{k}_3$ for each considered line contingency. The processes of learning would remain the same. The process of solving both the original CCGA with Algorithm 1 (to generate training instances) and the FR-CCGA (48)–(49) would remain similar, but with a larger set of contingency states and their respective contingency variables. This would lead to a sublinear increase in the data for the problem (with as many new preprocessed structures $\mathbf{K}_0$, $\mathbf{K}_1$, $\mathbf{k}_2$, and $\mathbf{k}_3$ as the number of line contingencies). This would not present a significant technical obstacle since the structure of the SCOPF problem would remain the same.

Notice that, in a broader perspective, the ML models of this work "*learn to solve an optimization problem*" while the post-processing feasibility recovery phase (Section V) precisely uses the same initial decomposition scheme to ensure feasibility. In this context, as long as a SCOPF model possesses a reasonable decomposition scheme, it is a candidate to the proposed general framework. Hence, the most relevant technical limitation for this framework is the development of decomposition approaches for the underlying optimization problem. As future research, security-constrained versions of convexifications of the AC OPF, such as those applied in [16] and [17], will be studied instead of the preventive DC-SCOPF.

VIII. CONCLUSION

This paper proposed a tractable methodology that combines deep learning models and robust optimization for generating solutions for the preventive DC-SCOPF problem. The considered SCOPF modeled generator contingencies and the automatic primary response of synchronized units. Computational results over two large test cases demonstrate the practical relevance of the methodology as a scalable, easy to specify, and cost-efficient alternative tool for managing short-term scheduling.

REFERENCES

[1] O. Alsac and B. Stott, "Optimal load flow with steady-state security," *IEEE Trans. Power App. Syst.*, vol. PAS-93, no. 3, pp. 745–751, May 1974.

[2] F. Bouffard, F. D. Galiana, and J. M. Arroyo, "Umbrella contingencies in security-constrained optimal power flow," in *Proc. 15th Power Syst. Comput. Conf.*, Liège, Belgium, 2005.

[3] Y. Li and J. D. McCalley, "Decomposed SCOPF for improving efficiency," *IEEE Trans. Power Syst.*, vol. 24, no. 1, pp. 494–495, Feb. 2009.

[4] F. Capitanescu *et al.*, "State-of-the-art, challenges, and future trends in security constrained optimal power flow," *Elect. Power Syst. Res.*, vol. 81, no. 8, pp. 1731–1741, Aug. 2011.

[5] Q. Wang, J. D. McCalley, T. Zheng, and E. Litvinov, "Solving corrective risk-based security-constrained optimal power flow with Lagrangian relaxation and benders decomposition," *Int. J. Elec. Power*, vol. 75, pp. 255–264, Feb. 2016.

[6] Y. Dvorkin, P. Henneaux, D. S. Kirschen, and H. Pandžić, "Optimizing primary response in preventive security-constrained optimal power flow," *IEEE Syst. J.*, vol. 12, no. 1, pp. 414–423, Mar. 2016.

[7] M. Velay, M. Vinyals, Y. Besanger, and N. Retiere, "Fully distributed security constrained optimal power flow with primary frequency control," *Int. J. Elec. Power*, vol. 110, pp. 536–547, Sep. 2019.

[8] A. Street, F. Oliveira, and J. M. Arroyo, "Contingency-constrained unit commitment with $n-k$ security criterion: A robust optimization approach," *IEEE Trans. Power Syst.*, vol. 26, no. 3, pp. 1581–1590, Aug. 2011.

[9] A. Khodaei and M. Shahidehpour, "Security-constrained transmission switching with voltage constraints," *Int. J. Electr. Power Energy Syst.*, vol. 35, no. 1, pp. 74–82, 2012.

[10] A. Moreira, A. Street, and J. M. Arroyo, "An adjustable robust optimization approach for contingency-constrained transmission expansion planning," *IEEE Trans. Power Syst.*, vol. 30, no. 4, pp. 2013–2022, Jul. 2015.

[11] A. Velloso, P. Van Hentenryck, and E. S. Johnson, "An exact and scalable problem decomposition for security-constrained optimal power flow," in *Proc. 21st Power Syst. Comput. Conf.*, 2020.

[12] Z. Zhou, T. Levin, and G. Conzelmann, "Survey of u.s. ancillary services markets," Argonne Nat. Lab., Argonne, IL (United States), Tech. Rep. ANL/ESD-16/1, Jan. 2016.

[13] L. Platbrood, F. Capitanescu, C. Merckx, H. Crisciu, and L. Wehenkel, "A generic approach for solving nonlinear-discrete security-constrained optimal power flow problems in large-scale systems," *IEEE Trans. Power Syst.*, vol. 29, no. 3, pp. 1194–1203, May 2014.

[14] F. Capitanescu, "Critical review of recent advances and further developments needed in ac optimal power flow," *Electr. Power Syst. Res.*, vol. 136, pp. 57–68, Jul. 2016.

[15] F. Li and R. Bo, "Dcopf-based lmp simulation: Algorithm, comparison with ACOPF, and sensitivity," *IEEE Trans. Power Syst.*, vol. 22, no. 4, pp. 1475–1485, Nov. 2007.

[16] C. Coffrin and P. Van Hentenryck, "A linear-programming approximation of ac power flows," *INFORMS J. Comput.*, vol. 26, no. 4, pp. 718–734, 2014.

[17] C. Coffrin, H. L. Hijazi, and P. Van Hentenryck, "The QC relaxation: A theoretical and computational study on optimal power flow," *IEEE Trans. Power Syst.*, vol. 31, no. 4, pp. 3008–3018, Jul. 2016.

[18] B. Eldridge, R. P. O'Neill, and A. Castillo, "Marginal loss calculations for the dcopf," *Federal Energy Regulatory Commission*, Tech. Rep., 2017.

[19] A. Marano-Marcolini, F. Capitanescu, J. L. Martinez-Ramos, and L. Wehenkel, "Exploiting the use of DC SCOPF approximation to improve iterative AC SCOPF algorithms," *IEEE Trans. Power Syst.*, vol. 27, no. 3, pp. 1459–1466, Aug. 2012.

[20] E. Lannoye, D. Flynn, and M. O'Malley, "Transmission, variable generation, and power system flexibility," *IEEE Trans. Power Syst.*, vol. 30, no. 1, pp. 57–66, Jan. 2015.

[21] P. Kundur, N. J. Balu, and M. G. Lauby, Power System Stability and Control. New York: McGraw-Hill, 1994, vol. 7,.

[22] J. F. Restrepo and F. D. Galiana, "Unit commitment with primary frequency regulation constraints," *IEEE Trans. Power Syst.*, vol. 20, no. 4, pp. 1836–1842, Nov. 2005.

[23] K. Karoui, H. Crisciu, and L. Platbrood, "Modeling the primary reserve allocation in preventive and curative security constrained OPF," in *Proc. IEEE PES Trans. Distrib. Conf. Expo.*, Apr. 2010, pp. 1–6.

[24] X. A. Sun, "Robust optimization in electric power systems," Advances and Trends in Optimization With Engineering Applications, T. Terlaky, M. F. Anjos, and S. Ahmed, Eds., 2017, pp. 357–365.

[25] D. Bertsimas, E. Litvinov, X. A. Sun, J. Zhao, and T. Zheng, "Adaptive robust optimization for the security constrained unit commitment problem," *IEEE Trans. Power Syst.*, vol. 28, no. 1, pp. 52–63, Feb. 2013.

[26] B. Zeng and L. Zhao, "Solving two-stage robust optimization problems using a column-and-constraint generation method," *Oper. Res. Lett.*, vol. 41, no. 5, pp. 457–461, Sep. 2013.

[27] H. Sasaki, M. Watanabe, J. Kubokawa, N. Yorino, and R. Yokoyama, "A solution method of unit commitment by artificial neural networks," *IEEE Trans. Power Syst.*, vol. 7, no. 3, pp. 974–981, Aug. 1992.

[28] Z. Ouyang and S. Shahidehpour, "A hybrid artificial neural network-dynamic programming approach to unit commitment," *IEEE Trans. Power Syst.*, vol. 7, no. 1, pp. 236–242, Feb. 1992.

[29] A. S. Xavier, F. Qiu, and S. Ahmed, "Learning to solve large-scale security-constrained unit commitment problems," *INFORMS J. Comput.*, 2019, *arXiv:1902.01697*.

[30] Z. Yang and S. Oren, "Line selection and algorithm selection for transmission switching by machine learning methods," in *Proc. IEEE Milan PowerTech.*, 2019, pp. 1–6.

[31] E. S. Johnson, S. Ahmed, S. S. Dey, and J.-P. Watson, "A k-nearest neighbor heuristic for real-time dc optimal transmission switching," 2020, *arXiv:2003.10565*.

[32] V. J. Gutierrez-Martinez, C. A. Cañizares, C. R. Fuerte-Esquivel, A. Pizano-Martinez, and X. Gu, "Neural-network security-boundary constrained optimal power flow," *IEEE Trans. Power Syst.*, vol. 26, no. 1, pp. 63–72, Feb. 2011.

[33] D. C. Costa, M. V. Nunes, J. P. Vieira, and U. H. Bezerra, "Decision tree-based security dispatch application in integrated electric power and natural-gas networks," *Electr. Power Syst. Res.*, vol. 141, pp. 442–449, Dec. 2016.

[34] S. Misra, L. Roald, and Y. Ng, "Learning for constrained optimization: Identifying optimal active constraint sets," 2018, *arXiv:1802.09639*.

[35] F. Fioretto, T. W. Mak, and P. Van Hentenryck, "Predicting AC optimal power flows: Combining deep learning and lagrangian dual methods," in *Proc. 34th Conf. Artif. Intell.*, (AAAI), 2020, pp. 630–637.

[36] A. J. Ardakani and F. Bouffard, "Identification of umbrella constraints in dc-based security-constrained optimal power flow," *IEEE Trans. Power Syst.*, vol. 28, no. 4, pp. 3924–3934, Nov. 2013.

[37] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

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

Alexandre Velloso received the M.S. degree in mathematical methods in finance from the Institute for Pure and Applied Mathematics, Rio de Janeiro, Brazil, and the Ph.D degree in electrical engineering from the Pontifical Catholic University, also in Rio de Janeiro, Brazil. He has been a Permanent Staff Member with Finep, the Brazilian Innovation Agency since 2010. He is currently a Visiting Research Scholar with the H. Milton Stewart School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta. His main research interests include optimization and machine learning applications to power systems.

Pascal Van Hentenryck (Member, IEEE) is the A. Russell Chandler III Chair and Professor, and the Associate Chair for Innovation and Entrepreneurship with the H. Milton Stewart School of Industrial and Systems Engineering, the Georgia Institute of Technology. His current research focuses on optimization and machine learning for energy and transportation systems. Several of his optimization systems, including the CHIP and OPL systems, have been in commercial use for more than 20 years.