UNSUPERVISED OPTIMAL POWER FLOW USING GRAPH NEURAL NETWORKS

Damian Owerko, Fernando Gama, Alejandro Ribeiro

Department of Electrical and Systems Engineering, University of Pennsylvania, PA, USA

ABSTRACT

Optimal power flow is a critical optimization problem that allocates power to the generators in order to satisfy the demand at a minimum cost. This is a non-convex problem shown to be NP-hard. We use a graph neural network to learn a nonlinear function between the power demanded and the corresponding allocation. We learn the solution in an unsupervised manner, minimizing the cost directly. To consider the power system constraints, we propose a novel barrier method that is differentiable and works on initially infeasible points. We show through simulations that the use of graph neural networks in this unsupervised learning context leads to solutions comparable to standard solvers while being computationally efficient and avoiding constraint violations.

Index Terms— optimal power flow, unsupervised learning, graph neural networks, graph signal processing

1. INTRODUCTION

Optimal power flow (OPF) is a critical optimization problem for the energy industry used to allocate electricity generation throughout the day, establish day-ahead market prices, and plan for grid infrastructure [1]. The objective of OPF is to minimize the cost of generating electrical power, subject to constraints imposed by the grid infrastructure, physical laws, and demand patterns. OPF is non-convex due to the sinusoidal nature of the alternating current and voltage and the constraints imposed by electrical interconnections of the grid [2]. In fact, OPF has been shown to be NP-hard [3]. Therefore, solving OPF exactly is computationally expensive [1].

A simple way to address OPF non-convexity is to solve a linear surrogate based on small-angle approximations. However, power grids are typically highly loaded, thus violating the small-angle approximation [4]. Developing more robust convex approximations is an area of active research. Examples include quadratic [5,6], second order conic [7] and semi-definite programming (SDP) [8,9] relaxations. On some topologies, these relaxations are exact, but may produce infeasible or suboptimal solutions for meshed topologies [10].

Recently, there was a resurgence of research into solving OPF exactly. The Grid Optimization (GO) Competition demonstrated the applicability of interior point methods to SC-OPF, a more challenging variant of OPF with additional constraints and mixed integer variables [11]. Competitors used interior point methods to solve SC-OPF with up to 32,000 nodes and within industry relevant timeframes [12]. However, no one approach was able to perform robustly across all real-world scenarios [12].

Data-driven approaches are another area of active research, with deep learning gaining particular attention. Inference of deep learning models is typically much faster than using traditional solvers. The feasibility of a candidate solution can be quickly verified or used to hot-start a more computationally expensive solver [13]. Imputation learning is used to replicate solutions obtained using interior point methods, but early approaches frequently violated constraints [14,15]. Feasible solutions can be encouraged by adding constraint violation penalty functions [16,17]. Deep learning models for OPF can also be trained unsupervised by minimizing the objective directly and enforcing constraints with penalties or dual methods [18,19]. This avoids the need for a computationally expensive labeled dataset and can yield solutions that are more cost optimal.

Another avenue of active research are topology adaptive learning approaches [20], which often leverage graph neural networks (GNNs) [21–23]. In these approaches, the system topology is an input to the model. This allows one model to produce solutions for many systems without retraining. GNNs are especially promising because both theoretical and empirical evidence indicates that they naturally respect the underlying graph [21,24].

This paper proposes a novel approach that lies at the intersection of unsupervised learning and topology adaptive methods. We recast the OPF problem as a graph signal processing task and utilize a graph neural network to solve it. GNN training is unsupervised and introduces a novel penalty function based on the log-barrier method.

2. OPTIMAL POWER FLOW

An electrical grid is an interconnected system that generates electricity and delivers it to consumers. OPF is concerned with minimizing the generation cost, while satisfying the electrical constraints of the grid [4]. Systems are typically modeled as a network of buses connected to each other by branches. Power can be produced or consumed at buses and flow between them via branches.

The main variables in OPF are the voltage, generated power, and load at each bus. Denote them as $v, s^g, s^d \in C^N$, respectively, assuming there are $N$ buses. Hence, the objective of OPF is to find the voltage $v$ and power generated $s^g$ that satisfies demand $s^d$ while minimizing the total production cost $C(s^g)$ and satisfying system constraints. The power generated and voltage at each node are constrained, which can be expressed as:

$$s^g_{\text{min}} \leq s^g \leq s^g_{\text{max}}$$  \hspace{1cm} (1)

$$v_{\text{min}} \leq |v| \leq v_{\text{max}}$$  \hspace{1cm} (2)

where $s^g_{\text{min}}, s^g_{\text{max}} \in C^N$, $v_{\text{min}}, v_{\text{max}} \in R^N$ and $\leq$ is a generalized inequality.

Branches connect different buses with each other, allowing power to flow from one bus to another. In this paper, we consider a generalized $n$-section branch model [25,26] as depicted by Figure 1. Let $f_I \in C^M$ be the power flowing out of the “from” buses and $f_I \in C^M$ denote the power flowing out of the “to” buses. $f_I$ and $f_I$ are determined by the voltages on either side of the branch:

$$f_I = (y + y_f) |v_f|^2 - y v_f v_f$$  \hspace{1cm} (3)

$$f_I = (y + y_f) |v_i|^2 - y v_i v_i$$  \hspace{1cm} (4)
where \( t \in \mathbb{C}^{\mathcal{N}} \) are the transformer ratios, \( y \in \mathbb{C}^{\mathcal{N}} \) are the line admittances, \( y'_{f} \in \mathbb{C}^{\mathcal{N}} \) are the from-side shunt admittances, and \( y''_{t} \in \mathbb{C}^{\mathcal{N}} \) are the branch-to-side shunt admittances. \( \mathbf{v}_{f} = C_{f} \mathbf{v} \) and \( \mathbf{v}_{t} = C_{t} \mathbf{v} \) are the voltages on either side of the branches where \( C_{f} \) and \( C_{t} \) are \( M \times N \) binary matrices relating the indices of the branches to the “from” and “to” buses, respectively. There is a limit \( f_{\text{max}} \in \mathbb{R}^{M} \) on the amount of power flowing through a branch:

\[
|f_{f}| \leq f_{\text{max}} \quad (5)
\]

\[
|f_{t}| \leq f_{\text{max}}. \quad (6)
\]

Branches have limits \( \theta_{\text{min}}, \theta_{\text{max}} \in \mathbb{R}^{M} \) on the phase angle difference between voltages on either side of each branch:

\[
\theta_{\text{min}} \leq \angle(\mathbf{v}_{f} \mathbf{v}_{t}) \leq \theta_{\text{max}} \quad (7)
\]

where \( \angle(\cdot) \) denotes the phase of a complex value. Finally, the net power flowing into a node equals the power flowing out of a node:

\[
\mathbf{s}^{g} - \mathbf{s}^{d} = (\mathbf{y}^{g})^{*} |\mathbf{v}|^{2} + C_{f}^{T} \mathbf{f}_{f} + C_{t}^{T} \mathbf{f}_{t}. \quad (8)
\]

where \( \mathbf{y}^{g} \in \mathbb{C}^{\mathcal{N}} \) describes the admittance from the buses to the ground.

3. GRAPH NEURAL NETWORKS

3.1. Graph Signal Processing

Graph signal processing (GSP) has emerged as a convenient framework to describe, analyze and solve problems that are distributed in nature [27]. Let \( \mathcal{G} = (\mathcal{B}, \mathcal{E}, \mathcal{W}) \) be a graph where \( \mathcal{B} = \{1, \ldots, \mathcal{N}\} \) is the set of \( N \) nodes, \( \mathcal{E} \subseteq \mathcal{B} \times \mathcal{B} \) is the set of edges, and \( \mathcal{W} : \mathcal{E} \to \mathbb{R} \) is a weight function that assigns a scalar to each edge. Data is described as a signal \( \mathbf{z} : \mathcal{B} \to \mathbb{R}^{F} \) on the nodes with \( F \) measurements or features. It is often convenient to represent a graph signal as a matrix, \( \mathbf{Z} \in \mathbb{R}^{N \times F} \), where the \( i^{th} \) row is \( \mathbf{z}(i) \).

A graph shift operator (GSO) captures the underlying graph support. A real symmetric matrix \( \mathbf{A} \in \mathbb{R}^{N \times N} \) is a GSO if its elements satisfy \( |\mathbf{A}|_{ij} = 0 \) whenever \( (i, j) \notin \mathcal{E} \). The most commonly used GSOs include the adjacency matrix, the Laplacian matrix, and their normalized versions [27,28]. A GSO can be used to construct graph convolutional filters. A multiple-input multiple-output graph filter \( \mathbf{H}: \mathbb{R}^{N \times F} \to \mathbb{R}^{N \times F'} \) can be represented as a \( K^{th} \) order polynomial,

\[
\mathbf{H}(\mathbf{Z}; \mathbf{A}) = \sum_{k=0}^{K} \mathbf{A}^{k} \mathbf{ZH}_{k}, \quad (9)
\]

with matrix weights \( \mathbf{H}_{k} \in \mathbb{R}^{F \times F'} \) where \( F \) and \( F' \) is the number of input and output features, respectively.

3.2. Graph Neural Networks

A graph neural network (GNN) is a nonlinear map \( \Phi(\mathbf{Z}; \mathcal{H}, \mathbf{A}) \) that is applied to an input \( \mathbf{Z} \) and depends on the underlying graph \( \mathcal{G} \) via a GSO \( \mathbf{A} \). A GNN consists of a cascade of \( L \) layers, each of them a graph filter (9) followed by a point-wise non-linearity \( \sigma_{\ell} \).

\[
\Phi(\mathbf{Z}; \mathcal{H}, \mathbf{A}) = \mathbf{Z}_{L}, \quad \mathbf{Z}_{\ell} = \sigma_{\ell}(\mathbf{H}_{\ell}(\mathbf{Z}_{\ell-1}; \mathbf{A})) \quad (10)
\]

for \( \ell = 1, \ldots, L \), where \( \mathbf{Z}_{0} = \mathbf{Z} \) is the input signal [29]. The output \( \mathbf{Z}_{L} \in \mathbb{R}^{N \times F_{L}} \) of layer \( \ell \) is a graph signal with \( F_{\ell} \) features. The output of the last layer, \( \mathbf{Z}_{L} \), is the output of the GNN. The specific non-linearity \( \sigma_{\ell} \), the number of features \( F_{\ell} \) and the number of filter weights \( K_{\ell} \) are design choices. The filter weights \( \mathcal{H} = \{\mathbf{H}_{\ell} \in \mathbb{R}^{F_{\ell-1} \times F_{\ell}}, k = 0, \ldots, K_{\ell}, \ell = 0, \ldots, L\} \) are model parameters to be learned from data by a training process.

3.3. Optimal power flow as a graph signal processing problem

Since the OPF is a problem distributed on the topology of the power system, it can be described as a GSP task. Represent the power system buses by a set \( \mathcal{B} \) and its branches by a set \( \mathcal{E} \subseteq \mathcal{B} \times \mathcal{B} \). Hence, define \( \mathcal{G} \) to be a weighted graph with nodes \( \mathcal{B} \) and edges \( \mathcal{E} \subseteq \mathcal{E} \). It has edge weights \( \mathcal{W}(i, j) := w_{ij} \) for each \( (i, j) \in \mathcal{E} \), which depend on the admittance \( Y_{ij} \) of the transmission line:

\[
w_{ij} := \exp(-\alpha/|Y_{ij}|^{2}) \quad (11)
\]

where \( \alpha \) is a scaling factor. Neglect branches whose weight is less than a threshold \( \beta \), so that \( \mathcal{E}' := \{ (i, j) \in \mathcal{E} : w_{ij} > \beta \} \). The values of \( \alpha \) and \( \beta \) are hyperparameters. Denote by \( \mathbf{A} \in \mathbb{R}^{N \times N} \) the adjacency matrix of \( \mathcal{G} \) such that the elements of \( \mathbf{A} \), \( A_{ij} = w_{ij} \) if \( (i, j) \in \mathcal{E} \) and zero otherwise. Since the graph is undirected, the matrix \( \mathbf{A} \) is symmetric. By construction, it is a GSO on \( \mathcal{G} \). For simplicity, we ignore the line charging admittance \( Y_{ij} \), assuming that it is negligible [25].

The OPF problem variables can be reinterpreted as multifeatured graph signals. Let \( \mathbf{Z} \in \mathbb{R}^{N \times F} \) be a signal describing the inputs:

\[
\mathbf{Z} := \begin{bmatrix} \text{Rect}(s^{g}) & \text{Rect}(s_{\text{min}}^{g}) & \text{Rect}(s_{\text{max}}^{g}) & \mathbf{v}_{\text{min}} & \mathbf{v}_{\text{max}} \end{bmatrix} \quad (12)
\]

where \( \text{Rect}(\mathbf{z}) := \begin{bmatrix} \text{Re}(\mathbf{z}) & \text{Im}(\mathbf{z}) \end{bmatrix} \) represents complex vectors from \( \mathbb{C}^{N} \) as a real matrix in \( \mathbb{R}^{N \times 2} \) containing the real and imaginary component. Similarly, let \( \mathbf{X} \in \mathbb{R}^{N \times 4} \) be a graph signal describing the outputs of the OPF problem:

\[
\mathbf{X} := \begin{bmatrix} \text{Rect}(s^{d}) & \text{Rect}(\mathbf{v}) \end{bmatrix}. \quad (13)
\]

Notice that \( \mathbf{Z} \) describes both the power demanded at each node and the nodal inequality constraints on power generation and voltage. Including the constraints in the input signals is necessary because GNNs are permutation equivariant.

The OPF solution is feasible and minimizes the cost function,

\[
\arg\min_{\mathbf{X}(\mathbf{Z}) \in \mathcal{X}(\mathbf{Z})} C(\mathbf{X}) \quad (14)
\]

where \( C(\mathbf{X}) := C(s^{g}) \) and \( \mathcal{X}(\mathbf{Z}) \) is the feasible set—the set of output signals that satisfy the OPF constraints from Section 2. As explained in the introduction, finding the solution to (14) is difficult. Instead, we take a machine learning-based approach by finding a map \( \Phi(\mathbf{Z}) \). This map should minimize the expected value of the cost function, given some unknown distribution of input signals, \( \mathbf{Z} \sim \mathcal{D} \):

\[
\arg\min_{\Phi(\mathbf{Z})} \mathbb{E}_{\mathcal{D}}[C(\Phi(\mathbf{Z}))] \quad (15)
\]
Solving (15) in its generality is typically intractable [30]. Therefore, we choose a parametrization of the map \( \Phi(Z) = \Phi(Z; H) \) where \( \Phi \) now becomes a known family of functions parameterized by \( H \). We assume the existence of a dataset \( T = [Z] \) that can be used to approximate the expectation operator [31], giving rise to the well-studied empirical risk minimization (ERM) problem [31]. The solution to the ERM problem is a set of parameters,

\[
H^* = \arg\min_H \sum_{Z \in T} C(\Phi(Z)),
\]

that minimizes the average cost over the dataset.

The desirable properties of locality and scalability can be achieved by a careful choice of the model \( \Phi(Z; H) \). We focus on GNNs, as described in Section 3.2, to exploit their stability properties [24] that provide scalability [32, 33]. If we choose the above defined adjacency matrix \( A \) as the GSO, the model becomes \( \Phi(Z) = \Phi(Z; H, A) \).

4. UNSUPERVISED LEARNING WITH PENALTY FUNCTIONS

Finding a GNN parametrization (16) to the OPF problem is challenging because it is a non-convex constrained optimization problem. Instead of solving the problem directly, we can approximate the constraints using penalty functions. To illustrate this, consider that the optimal power flow problem can be expressed in standard form,

\[
\begin{aligned}
\min \quad & C(X) \\
\text{s.t.} \quad & g_i(X) \leq 0 \quad i = 1, ..., n \\
& h_i(X) = 0 \quad i = 1, ..., m
\end{aligned}
\]

where \( g_i \) and \( h_i \) are (non-convex) functions that capture all the constraints described in Section 2.

Instead of solving (16) directly, we formulate a related unconstrained optimization problem. Each inequality constraint \( g_i(X) \leq 0 \) is handled by adding a penalty function \( \phi_i(X) \) to the objective. Similarly, the equality constraints are replaced by \( \psi_i(X) \). The loss function, \( L(X) \), combines the cost function and the penalty functions,

\[
L(X) = C(X) + \sum_{i=1}^{n} \lambda_i \phi_i(X) + \sum_{i=1}^{m} \mu_i \psi_i(X)
\]

where \( \lambda_i, \mu_i \geq 0 \) are weight parameters. This can be used to define an unconstrained ERM problem:

\[
\hat{H} = \arg\min_H \sum_{Z \in T} L(\Phi(Z; H, A)).
\]

If \( \phi_i, \psi_i \) and \( C(X) \) are differentiable, then (19) can be solved using gradient descent.

Unlike \( H^* \) in (16), the solution of (19), \( \hat{H} \), is no longer guaranteed to produce feasible outputs \( \Phi(Z; \hat{H}) \in \mathcal{X}(Z) \) for all \( Z \in T \). This is the trade-off we make by approximating the problem with penalty functions. Nevertheless, determining the feasibility of a candidate solution is inexpensive. When the output of the model is not feasible, we can instead fall back on interior point methods, possibly using the output to hot-start the solver.

4.1. Choice of penalty functions

Our choice of the inequality penalty, \( \phi_i(X) \), is inspired by the log-barrier method [34]. In the log-barrier method, a logarithmic function approximates the inequality constraints. The log-barrier function has the form \(- (1/t) \log(-g_i(X)) \) where \( t > 0 \) is a parameter. Higher values of \( t \) provide finer approximations to the indicator function. The log-barrier function requires the output to be feasible, \( X \in \mathcal{X}(Z), \) otherwise the logarithm’s output is undefined. This is impractical for GNN training because the parameters \( H \) are randomly initialized and therefore, at the beginning of training, \( \Phi(Z; H) \) is unlikely to be feasible.

To overcome this, we define the extended logarithm, \( \log_s \), which is a linear piece-wise extension of the logarithm to \( \mathbb{R}_- \):

\[
\log_s(u) := \begin{cases} 
\log(u) & \text{if } (u \geq 1/s) \\
\log((u - \frac{1}{s}) + \log(\frac{1}{s}) & \text{otherwise}
\end{cases}
\]

where \( s \in \mathbb{R} \) is a parameter and the function’s maximum derivative. Hence, we define the inequality penalty function:

\[
\phi_i(X) := -(1/t) \log_s(-g_i(X)).
\]

Unlike the traditional log-barrier function, it provides a soft barrier, but converges to the log-barrier as \( s \) becomes large. Meanwhile, we use the square function as the penalty for the equality constraints:

\[
\psi_i(X) := h_i(X)^2.
\]

5. NUMERICAL EXPERIMENTS

To evaluate the efficacy of the proposed approach, we ran simulations on the IEEE-30 and IEEE-118 power systems from MATPOWER [25]. We use the following methodology to construct a dataset for each network. The test cases provide reference values for complex power demanded at each bus \( s^d \in \mathbb{C}^N \). We sample elements-wise from a uniform distribution around the reference load, following the same methodology as in [14],

\[
s^d \sim \text{Uniform}(0.9s^d_{\text{ref}}, 1.1s^d_{\text{ref}}).
\]

For both IEEE-30 and IEEE-118 systems, we generated 100,000 training samples and 1,000 test samples. 5% of training samples were reserved for validation. For each sample from the test set, we solved the corresponding OPF problem using the interior point solver called IPOPT [26, 35, 36]. The resulting ground truth solutions were not used in training; they served as a benchmark to compare performance of the GNN against solving OPF with IPOPT.

For each power system, we trained a GNN with \( L = 2 \) layers, \( K = 8^\text{th} \) order filters, \( F = 32 \) features at each hidden layer, and a step size \( \eta = 1 \times 10^{-4} \). The model was trained for 1000 epochs at a batch size of 256 with penalty function parameters \( s = 10 \) and \( t = 500 \). The hyperparameters were determined after a grid search with 484 combinations of \( K, F, L, \eta, s, t \). The aforementioned values minimized the validation loss on the IEEE-30 dataset. The adjacency matrix is constructed following Section 3.3 using \( \beta = 0.01 \). Meanwhile, \( \alpha \) was chosen so that the mean edge weight of the adjacency matrix was 0.5.

5.1. IEEE-30

The results indicate that the GNN can achieve a lower power generation cost than the IPOPT solution. On average, the cost of the GNN
Fig. 2: GNN performance on the IEEE-30 test dataset. (a) Distribution of GNN solution cost expressed as a fraction of the IPOPT solution cost excluding samples with constraint violations. (b) Distribution of the maximum constraint violation on each sample.

Fig. 3: GNN performance on the IEEE-30 test dataset. (a) Distribution of GNN solution cost expressed as a fraction of the IPOPT solution cost excluding samples with constraint violations. (b) Distribution of the maximum constraint violation on each sample.

solution is 4.142. Meanwhile, the IPOPT cost is 2.680% greater, at 4.253. The GNN produces feasible solutions in 24.66% of test samples. Figure 2a shows that across all samples when the GNN solution is feasible, it also has a lower cost than IPOPT.

On the other hand, in at 75.44% of the test samples, at least one constraint is violated. On average, 1.276% of constraints are violated; constraints on individual buses and branches are counted separately. Out of all the equality and inequality constraints in Section 2, only constraints on the voltage magnitudes (2) and line power flows (5) are violated. The severity of the violations is low. To quantify constraint violation severity, we divide the magnitude by which an inequality constraint is violated by the difference between the upper and lower bounds. Figure 2b shows that the normalized violation magnitude are always below 5% and are more concentrated around zero. This suggests that violations are due to the shape of the loss function or a “duality-gap”, rather than a failure of the GNN to express the constraints. With more fine-tuning of penalty function parameters, violations could be further minimized.

5.2. IEEE-118

Overall, the GNNs performance on the IEEE-118 system is substantially worse than on the IEEE-30 system. The GNN achieves an average cost of 903.96 compared to 836.10 for IPOPT. Figure 3a shows that the cost of the GNN output is consistently higher than the IPOPT solution. Constraint violations occur in 82.5% of the samples, but the only violated constraints are on the voltage magnitudes (2) and generator outputs (1). Moreover, as Figure 3b shows, constraint violations are more severe than on the IEEE-30 system.

There are two likely reasons for the comparatively worse performance on the IEEE-118 dataset. First, the hyperparameters were based on GNN performance on the IEEE-30 system. Second, and perhaps more importantly, the IEEE-118 system is qualitatively different: the case file defines unrealistically high-power flow limits [37]. This makes this case particularly difficult to solve using GNNs. Without a limit on the amount of power that can be transmitted between buses, the problem becomes non-local. However, the GNN architecture is designed around local exchanges of information [29] and struggles with such problems [38]. If this is the root cause of the poor performance, it would be a major limitation to the applicability of GNNs to OPF. Even when power flow limits are realistic, they may not be binding at low loads. Increasing the GNN depth could help performance. However, training deeper GNNs is an area of active research due to problems such as over-smoothing at large depths [39].

6. CONCLUSIONS

In this paper, we propose a novel approach to solve optimal power flow using graph neural networks and penalty functions. Our experiments suggest that, in the right conditions, graph neural networks can produce feasible solutions that are closer to optimum than from interior point methods. Specifically, when current or power rate limits are binding. At times of peak demand, this is the case with many real power systems, where transmission capacity is limited. Additionally, tuning the weights and parameters of the penalty functions remains a challenge.

We believe this initial work can potentially spark further interest in the use of graph neural networks in the toolbox of optimal power flow solutions. Based on works in other applications, graph neural networks could offer not only scalability, but also the capacity to reuse the same model on different systems without or with limited retraining. This may require revisiting the specification of power system test cases and creating benchmark datasets and benchmarks to facilitate comparing approaches.

7. REFERENCES

[1] M. B. Cain, R. P. O’Neil, and C. Anya, “History of Optimal Power Flow and Formulations;" pp. 1–31, 2012.
[2] B. Lesieutre and I. Hiskens, “Convexity of the set of feasible injections and revenue adequacy in FTR markets," IEEE Transactions on Power Systems, vol. 20, no. 4, pp. 1790–1798, Nov. 2005.
[3] D. Bienstock and A. Verma, “Strong NP-hardness of AC power flows feasibility," Operations Research Letters, vol. 47, no. 6, pp. 494–501, Nov. 2019.
[4] S. Chatzivasileiadis, “Lecture Notes on Optimal Power Flow (OPF)," Nov. 2018, arXiv:1811.00943.
[5] H. Hijazi, C. Coffrin, and P. V. Hentenryck, “Convex quadratic relaxations for mixed-integer nonlinear programs in power systems," Mathematical Programming Computation, vol. 9, no. 3, pp. 321–367, Sept. 2017.
[6] K. Sundar, H. Nagarajan, S. Misra, M. Lu, C. Coffrin, and R. Bent, “Optimization-Based Bound Tightening using a Strengthened QC-Relaxation of the Optimal Power Flow Problem," Jan. 2019, arXiv:1809.04565.
[7] R. A. Jabr, “Radial distribution load flow using conic programming," IEEE Transactions on Power Systems, vol. 21, no. 3, pp. 1458–1459, Aug. 2006.
[8] X. Bai, H. Wei, K. Fujisawa, and Y. Wang, “Semidefinite programming for optimal power flow problems," International
[9] R. A. Jabr, “Exploiting sparsity in SDP relaxations of the OPF problem,” IEEE Transactions on Power Systems, vol. 27, no. 2, pp. 1138–1139, May 2012.

[10] 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.

[11] I. Aravena et al., “Recent Developments in Security-Constrained AC Optimal Power Flow: Overview of Challenges,” in the ARPA-E Grid Optimization Competition,” Operations Research, Sept. 2023.

[12] F. Safdarian et al., “Grid Optimization Competition on Synthetic and Industrial Power Systems,” in 2022 North American Power Symposium (NAPS), Oct. 2022, pp. 1–6.

[13] W. Dong, Z. Xie, G. Kestor, and D. Li, “Smart-PGSim: Using Neural Network to Accelerate AC-OPF Power Grid Simulation,” in SC20: International Conference for High Performance Computing, Networking, Storage and Analysis, Nov. 2020, pp. 1–15.

[14] N. Guha, Z. Wang, and A. Majumdar, “Machine Learning for AC Optimal Power Flow,” in ICML 2019, 2019.

[15] D. Owerko, F. Gama, and A. Ribeiro, “Optimal Power Flow Using Graph Neural Networks,” in ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), May 2020, pp. 5930–5934.

[16] X. Pan, T. Zhao, M. Chen, and S. Zhang, “DeepOPF: A deep neural network approach for security-constrained DC optimal power flow,” IEEE Transactions on Power Systems, vol. 36, no. 3, pp. 1725–1735, 2021.

[17] X. Pan, M. Chen, T. Zhao, and S. H. Low, “DeepOPF: A Feasibility-Optimized Deep Neural Network Approach for AC Optimal Power Flow Problems,” IEEE Systems Journal, vol. 17, no. 1, pp. 673–683, Mar. 2023.

[18] W. Huang and M. Chen, “DeepOPF-NGT: A Fast Unsupervised Learning Approach for Solving AC-OPF Problems without Ground Truth,” in Climate Change AI. July 2021, Climate Change AI.

[19] K. Chen, S. Bose, and Y. Zhang, “Unsupervised Deep Learning for AC Optimal Power Flow via Lagrangian Duality,” in GLOBECOM 2022 - 2022 IEEE Global Communications Conference, Dec. 2022, pp. 5305–5310.

[20] M. Zhou, M. Chen, and S. H. Low, “DeepOPF-FT: One Deep Neural Network for Multiple AC-OPF Problems With Flexible Topology,” IEEE Transactions on Power Systems, vol. 38, no. 1, pp. 964–967, Jan. 2023.

[21] S. Liu, C. Wu, and H. Zhu, “Topology-aware Graph Neural Networks for Learning Feasible and Adaptive AC-OPF Solutions,” IEEE Transactions on Power Systems, pp. 1–11, 2022.

[22] T. Pham and X. Li, “Reduced Optimal Power Flow Using Graph Neural Network,” June 2022, arXiv:2206.13591.

[23] T. Falconer and L. Mones, “Leveraging Power Grid Topology in Machine Learning Assisted Optimal Power Flow,” IEEE Transactions on Power Systems, vol. 38, no. 3, pp. 2234–2246, May 2023.

[24] F. Gama, J. Bruna, and A. Ribeiro, “Stability Properties of Graph Neural Networks,” IEEE Transactions on Signal Processing, vol. 68, pp. 5680–5695, 2020.