Learning to Solve the AC Optimal Power Flow via a Lagrangian Approach

Ling Zhang
Department of Electrical and Computer Engineering
University of Washington, Seattle WA, USA
lzhang18@uw.edu

Baosen Zhang
Department of Electrical and Computer Engineering
University of Washington, Seattle WA, USA
zhangbao@uw.edu

2022 North American Power Symposium (NAPS) | 978-1-6654-9921-7/22/$31.00 ©2022 IEEE | DOI: 10.1109/NAPS56150.2022.10012237

***Abstract*—Using deep neural networks to predict the solutions to AC optimal power flow (ACOPF) problems has been an active direction of research. However, because the ACOPF is nonconvex, it is difficult to construct a good data set that contains mostly globally optimal solutions. To overcome the challenge that the training data may contain suboptimal solutions, we propose a Lagrangian-based approach. First, we use a neural network to learn the dual variables of the ACOPF problem. Then, from the predicted dual variables, we use a second neural network to predict solutions of the partial Lagrangian and use the predicted solutions as warm starts for the ACOPF problem. We test our approach on standard and modified IEEE networks and show that our approach can reach more globally optimal solutions with significant computational speedup even when the training data consists of mostly suboptimal solutions.**

***Index Terms*—AC optimal power flow, deep learning, Lagrangian-based approach.**

I. Introduction

The AC optimal power flow (ACOPF) problems are fundamental to power system operations, but they are often computationally expensive to solve in real-time [1]. Recently, machine learning has emerged as a popular method to aid in solving ACOPFs [2]–[5]. By treating the optimization problem as a function that maps loads to the optimal solutions (voltage and angles), supervised learning techniques can be used to train a neural network (NN) that replaces a nonlinear programming solver [6]–[9]. Since training can be done offline using historical or simulated data, neural networks can be used in real-time to reduce computational burdens.

Despite the many advances in using learning to solve ACOPFs, there are still some unresolved challenges. The quality of the learned solutions is fundamentally limited by the quality of the training data sets. In the context of ACOPF, the training set consists of pairs of active/reactive loads and their corresponding ACOPF solutions. A key, and often unstated assumption, is that the solutions in the training set are the optimal ones. However, the training set is typically constructed using existing nonlinear programming solvers, and there is no guarantee that the solutions are in fact globally optimal.

Historically, the presence of multiple solutions to the ACOPF problem has been solved by assuming that there is only one "practical" solution (i.e., with high voltage) [10]. However, many recent works have pointed out that multiple solutions do occur under reasonable conditions and cannot be easily ruled out [11]–[13]. Therefore, it is not easy to reprocess the data set to rule out bad solutions.

978-1-6654-9921-7/22/$31.00 ©2022 IEEE

Related to the presence of suboptimal solutions, a very difficult setting for learning can arise when several solutions are associated with similar loads. Because of the presence of multiple solutions, a small change in load may lead a solver to jump between distinct solutions. Therefore, the training set could include data that are close in terms of the load, but quite different in the solutions. For a neural network trained using regression loss, it would output the average of the local solutions, leading to an initialization point that neither increase the computation speed nor helps with the quality of the solutions.

In this paper, we present a machine learning architecture that overcomes the challenge of multiple suboptimal local solutions in the training data set. Instead of focusing on the load/solution pairs, we can think of these machine learning methods to be producing a good *warm start* for a solver, and learning a good warm start would offer significant computational speedups and improvement on solution quality [14].

We first learn a neural network that maps load to the dual variable of the power balance constraints. These dual variables are the locational marginal prices and would be readily available from any modern nonlinear solver. These dual variables are used to form a partial Lagrangian, whose solution we also learn via a neural network. Then we use the predicted solution of the partial Lagrangian as a warm start [15]. Interestingly, this warm starting point tends to be closer to the globally optimal solution of the ACOPF, even if the training data set only has suboptimal solutions or a mixture of global and local solutions. Therefore, by using the learned warm start as an initialization point, we could have better solution quality than directly learning based on load/solution pairs.

We show that our duality-based approach outperforms existing approaches on modified IEEE 22-, 39-, and 118-bus networks [11]. The difference is especially significant if the training data set contains some strictly suboptimal solutions.

II. Problem Formulation

A. ACOPF Formulation

Consider a power system network where n buses are connected by m edges. For bus i, let V_i denote its voltage magnitude, θ_i its angle, P_i^G and Q_i^G the active and reactive

output of the generator and P_i^D and Q_i^D the active and reactive load. We use P_{ij}^f and Q_{ij}^f to denote the active and reactive power flowing from bus i to bus j. The admittance between buses i and j is $g_{ij} - jb_{ij}$. We use θ_{ij} as a shorthand for $\theta_i - \theta_j$.

The ACOPF problem is to minimize the cost of active power generations while satisfying a set of constraints [11]:

$$\min_{\mathbf{V},\boldsymbol{\theta}} \textstyle\sum_i c_i(P_i^G) \tag{1a}$$

$$\text{s.t. } P_i^G = P_i^D + \textstyle\sum_{j=1}^N P_{ij}^f \tag{1b}$$

$$Q_i^G = Q_i^D + \textstyle\sum_{j=1}^N Q_{ij}^f \tag{1c}$$

$$P_{ij}^f = V_i^2 g_{ij} - V_i V_j (g_{ij}\cos(\theta_{ij}) - b_{ij}\sin(\theta_{ij})) \tag{1d}$$

$$Q_{ij}^f = V_i^2 \hat{b}_{ij} - V_i V_j (b_{ij}\cos(\theta_{ij}) + g_{ij}\sin(\theta_{ij})) \tag{1e}$$

$$\underline{\mathbf{V}}_i \le V_i \le \bar{V}_i \tag{1f}$$

$$\underline{\mathbf{P}}_i^G \le P_i^G \le \bar{P}_i^G \tag{1g}$$

$$\underline{\mathbf{Q}}_i^G \le Q_i^G \le \bar{Q}_i^G \tag{1h}$$

$$(P_{ij}^f)^2 + (Q_{ij}^f)^2 \le (S_{ij}^{\max})^2 \tag{1i}$$

where $\hat{b}_{ij} = b_{ij} - 0.5 b_{ij}^C$ and b_{ij}^C is the line charging susceptance. The constraints (1b) and (1c) enforce power balance, (1d) and (1e) are the AC power flow equations, (1f) limits the bus voltage magnitudes, (1g) and (1h) represent the active and reactive limits and (1i) are the line flow limits.

The problem in (1) is nonconvex and can have multiple local solutions. More precisely, local solutions are all the solutions that satisfy local optimality conditions, for example, the KKT conditions or second order ones [14]. Out of this set, the solutions with the lowest cost are called the global ones. We sometimes refer to the local solutions that are not global as strict local solutions.

B. Challenges of Using Learning for ACOPF

Machine learning is often applied to the optimization problem in (1) by viewing it as the mapping from the demands P^D and Q^D to the solutions $\mathbf{V}$ and $\boldsymbol{\theta}$, with the goal of finding a proxy function to this mapping. Training data is generated by solving (1) for a number of demands and collecting the corresponding solutions. Several different learning architectures have been proposed, including direct regression [2], using sensitivity information [6], and emulation of an iterative solver [7]. The constraints in (1b) to (1i) are nontrivial to enforce, and an additional call of power flow or optimal power flow on a smaller problem are often used [7], [16].

However, existing learning methods face a common challenge, stemming from the fact that the ACOPF problem in (1) is nonconvex and may have multiple solutions [11]–[13]. A number of recent works have shown that for reasonable operation conditions, there could be multiple solutions that differ significantly in cost, but all have practical values (e.g., with voltages being all close to 1 p.u.) [17]–[20].

Many nonlinear programming (NLP) solvers have been developed for the ACOPF problem, and their speed and efficiency have improved dramatically (e.g., see [21] and the references within). But NLP solvers are typically only able to return one of the feasible solutions, and there is generally no way to tell whether these feasible solutions are globally optimal. Therefore, using the solutions returned by NLP solvers as ground truth data for training may fundamentally limit the performance of the learning algorithm.

The solution returned by an NLP solver is also sensitive to a variety of factors, and small changes in demand can lead to a large change in the solution. For example, small changes in demand can lead to the solution switching between global and local. Therefore, a data set created directly using the solutions returned by NLP solvers may very well consist of a mixture of local and global solutions, which tend to be very confusing for a learning algorithm.

III. Algorithm

In this section, we describe our learning approach to find more optimal solutions to ACOPF problems using neural networks, even when the training data set contains a mixture of local and global solutions. We first give a partial Lagrangian-based approach, then discuss the learning architecture. The key of the approach is to obtain global solutions from a training set of local ones.

A. Lagrangian-based Approach

In paper [15], we gave an iterative approach to improve the solution quality by alternatively solving (1) and its partial Lagrangian. We briefly review the algorithm here. The partial Lagrangian for (1) is formed by dualizing the active and reactive power balance constraints (1b) and (1c). Suppose the Lagrangian multipliers associated with (1b) and (1c) are μ_i^P and μ_i^Q, respectively, then the partial Lagrangian for (1) is:

$$\begin{aligned}\mathcal{L}_{\boldsymbol{\mu}}(\mathbf{V},\boldsymbol{\theta}) = &\textstyle\sum_i c_i(P_i^G) + \sum_i \mu_i^P (P_i^D + P_i^f - P_i^G) \\ &+ \sum_i \mu_i^Q (Q_i^D + Q_i^f - Q_i^G)\end{aligned} \tag{2a}$$

$$\text{s.t. } \underline{\mathbf{V}}_i \le V_i \le \bar{V}_i \tag{2b}$$

$$\underline{\mathbf{P}}_i^G \le P_i^G \le \bar{P}_i^G \tag{2c}$$

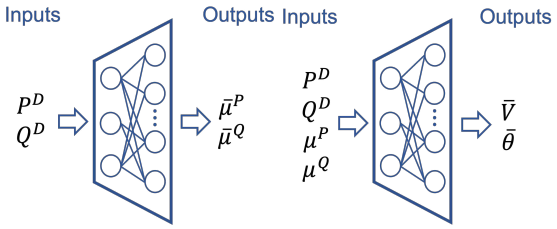
$$\underline{\mathbf{Q}}_i^G \le Q_i^G \le \bar{Q}_i^G \tag{2d}$$

$$(P_{ij}^f)^2 + (Q_{ij}^f)^2 \le (S_{ij}^{\max})^2 \tag{2e}$$

where $P_i^f = \sum_{j=1}^N V_i^2 g_{ij} - V_i V_j (g_{ij}\cos(\theta_{ij}) - b_{ij}\sin(\theta_{ij}))$, and $Q_i^f = \sum_{j=1}^N V_i^2 \hat{b}_{ij} - V_i V_j (b_{ij}\cos(\theta_{ij}) + g_{ij}\sin(\theta_{ij}))$ are the AC power flow equations.

It turns out the solution of the partial Lagrangian in (2) tends to be close to the global optimal solution of the original ACOPF in (1). This is true even if the multipliers μ^P and μ^Q are the ones associated with the strict local solutions (see [15] for more details). Therefore, the solution of (2) serves as a good warm start point for an ACOPF solver.

In this paper, we train two neural networks to replace explicitly solving (1) and (2). The first one predicts the dual variables of the active and reactive power balance constraints from the load, and the second one to predicts the solution of (2) from the load and the predicted multipliers using the



(a) The neural network to learn the dual variables of the power balance constraints.

(b) The neural network to predict the solutions of the partial Lagrangian.

Fig. 1: The two neural networks to be trained.

first neural network. The output of the second neural network is used as a warm start point for an ACOPF solver. Since this starting point is close to the global solution, the ACOPF solver is solved much faster and is more likely to find the global solution than a solver with a flat or random start.

B. Training of Neural Networks

The architectures of the two neural networks are shown in Fig. 1. The first neural network takes the active and reactive load demands $(\boldsymbol{P}^D, \boldsymbol{Q}^D)$ as the input, and the output is the predicted multipliers, denoted by $(\bar{\boldsymbol{\mu}}_P, \bar{\boldsymbol{\mu}}_Q)$. Let $\mathbf{x}$ be the collection of voltage magnitudes and angles, $\boldsymbol{\mu}$ be the collection of $(\boldsymbol{\mu}_P, \boldsymbol{\mu}_Q)$, and $(\mathbf{x}^i, \boldsymbol{\mu}^i)$ be the i-th pair of data in the training set, then the first neural network, denoted as g_{acopf} and parameterized by $\boldsymbol{w}$, is trained by minimizing:

$$\min_{\boldsymbol{w}} \frac{1}{N} \sum_{i=1}^{N} (\boldsymbol{\mu}^i - g_{acopf}(\mathbf{x}^i; \boldsymbol{w}))^2. \tag{3}$$

The second neural network emulates solving the partial Lagrangian in (2). The input for the second neural network is the active and reactive load demands $(\boldsymbol{P}^D, \boldsymbol{Q}^D)$, as well as the associated dual variable solutions $(\boldsymbol{\mu}_P, \boldsymbol{\mu}_Q)$. The output of the neural network is the solution to (2), denoted by $(\bar{\mathbf{V}}, \bar{\boldsymbol{\theta}})$. Let $\mathbf{z}$ be the collection of inputs, $\bar{\mathbf{x}}$ be the collection of outputs, and $(\mathbf{z}^i, \bar{\mathbf{x}}^i)$ be the i'th pair of data in the training set, then the second network, denoted by g_{dual} with weights $\boldsymbol{w}$ is trained by minimizing

$$\min_{\boldsymbol{w}} \frac{1}{N} \sum_{i=1}^{N} (\bar{\mathbf{x}}^i - g_{dual}(\mathbf{z}^i; \boldsymbol{w}))^2. \tag{4}$$

C. Making Predictions in Real-time

After training, we use the process in Fig. 2 to make predictions. We use the first trained neural network to predict the dual variables $(\bar{\boldsymbol{\mu}}_P, \bar{\boldsymbol{\mu}}_Q)$ from the input load. Then from the predicted dual variable solutions, we use the second trained neural network to predict the solutions $(\bar{\mathbf{V}}, \bar{\boldsymbol{\theta}})$ of the Lagrangian. Then we call the NLP solver to solve (1) using $(\bar{\mathbf{V}}, \bar{\boldsymbol{\theta}})$ as the initial point. This learning algorithm is summarized below as Algorithm 1.

In our approach, even when the training data set contains suboptimal solutions, the solution of the Lagrangian would be a good warm start that makes the NLP solver get around being

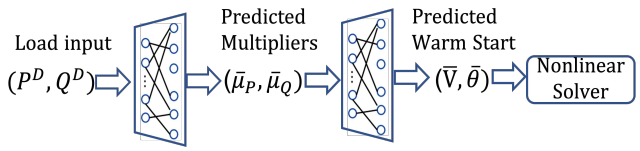


Fig. 2: Outline of the solution process.

Algorithm 1: Solving ACOPF using learning

Inputs: $(\mathbf{P}^D, \mathbf{Q}^D)$

First trained neural network to predict multipliers:

$g_{acopf}(\mathbf{P}^D, \mathbf{Q}^D; \boldsymbol{w}) \longrightarrow (\bar{\boldsymbol{\mu}}_P, \bar{\boldsymbol{\mu}}_Q)$

Second trained neural network to predict solutions to (2):

$g_{dual}(\mathbf{P}^D, \mathbf{Q}^D, \bar{\boldsymbol{\mu}}_P, \bar{\boldsymbol{\mu}}_Q; \boldsymbol{w}) \longrightarrow (\bar{\mathbf{V}}, \bar{\boldsymbol{\theta}})$

Call NLP solver for (1) initialized at $(\bar{\mathbf{V}}, \bar{\boldsymbol{\theta}})$;

Outputs: Solutions $(\hat{\mathbf{V}}, \hat{\boldsymbol{\theta}})$ to (1).

trapped at strictly local solutions. The reason is that the partial Lagrangian in (2) has "nice" geometry: It has minimums that are near the globally optimal solution of the ACOPF problem, even if it is formed with the multipliers obtained at local optimal solutions of the ACOPF problem [15].

Also, our algorithm is robust in the sense that the solutions $\bar{\mathbf{x}}$ to partial Lagrangian are not sensitive to the variations in $\bar{\boldsymbol{\mu}}$. That is, even if the multipliers $\bar{\boldsymbol{\mu}}$ associated with different local solutions are different, the resulting solutions to partial Lagrangian do not change much. This also means we do not ask the predictions of the neural network to be very accurate. Therefore, the training set for our algorithm need not be very large. We sketch the geometric intuitions behind our algorithm in Section IV. We validate our algorithm in standard and modified IEEE benchmark systems, and report simulation results in Section V.

IV. Geometry and Intuition

In this section, we use a 2-bus network as an example to shed some light on why Algorithm 1 might learn more globally optimal solutions, even when the training data consists of a lot of local solutions. In the 2-bus network, for simplicity, we ignore the reactive power and set both voltage magnitudes to be 1 per unit. Suppose bus 1 is a generator and also is the reference bus with an increasing cost function $c(\cdot)$, and bus 2 is the load bus with angle $-\theta$. The line admittance is $g - jb$. Given a load of l at bus 2 and ignoring all constraints except for the load balancing one, the ACOPF in (1) becomes

$$\min_{\theta} \; c(g - g\cos(\theta) + b\sin(\theta)) \tag{5a}$$

$$\text{s.t. } l + g - g\cos(\theta) - b\sin(\theta) = 0. \tag{5b}$$

This is an example of an OPF with a disconnected feasible space, since there are two distinct solutions to (5b) and we are asking for the lower cost one.

To see how an NLP solver would approach this problem, we adopt the common practice in nonlinear programming and

form a penalized version of (5) [14], [22]. The penalized unconstrained problem is given by

$$\begin{aligned}\mathcal{L}_\rho =& c(g - g\cos(\theta) + b\sin(\theta)) \\ &+ \frac{\rho}{2}(l + g - g\cos(\theta) - b\sin(\theta))^2,\end{aligned} \tag{6}$$

where ρ is a penalty parameter. For large enough ρ, the solutions of (6) would be very close to those of (5) [14]. The function $\mathcal{L}_\rho$ is plotted in Fig. 3. We can see that there are two local minima, with the left one being global. However, both minima satisfy first- and second-order optimality conditions. Therefore, if an NLP solver is initialized with a poor starting point, it would be stuck at the strict local solution.

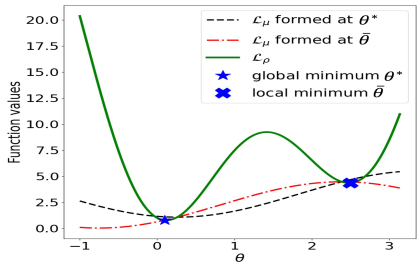


Fig. 3: Geometry of the penalized objective function $\mathcal{L}_\rho$ and the partial Lagrangian $\mathcal{L}_\mu$. The line admittance is $g - jb$ and the penalty parameter is 2. The red curve is the partial Lagrangian formed at the strict local solution and the black curve is formed at the global solution.

Now suppose μ is the multiplier corresponding to the equality constraint (5b) at *the strict local solution*. The partial Lagrangian of (5) by dualizing (5b) is:

$$\begin{aligned}\mathcal{L}_\mu =& c(g - g\cos(\theta) + b\sin(\theta)) \\ &+ \mu(l + g - g\cos(\theta) - b\sin(\theta)).\end{aligned} \tag{7}$$

Since the sinusoidal functions are periodic with period 2π, let us consider the range $\theta \in [-\pi, \pi]$. It is interesting now to compare the solution of $\mathcal{L}_\mu$ to the original problem in (5) (or equivalently, $\mathcal{L}_\rho$). The red curve in Fig. 3 plots $\mathcal{L}_\mu$ at the local minimum and the black curve at the global minimum. We can observe an interesting fact that the minimum of $\mathcal{L}_\mu$ is close to the global minimum of $\mathcal{L}_\rho$, even when the multiplier at the strict local solution is used.

It is instructive to think about a training data set with both local and global solutions for the same load, and compare the learned warm starts using direct regression and Algorithm 1. Suppose a regression method is used to minimize the distance between a predicted solution and the solutions in the training set. Since a mixture of local and global solutions are used in training, the learned neural network would make a prediction that is the average of the two solutions, as shown in Fig. 4. But it may be closer to the local solution rather than the global one. Preprocessing the training data may alleviate some of these issues, but that is likely to be cumbersome and removes some of the appeals of using machine learning.

When Algorithm 1 is used, we first predict the multipliers from the load. Since the training set is a mix of local and

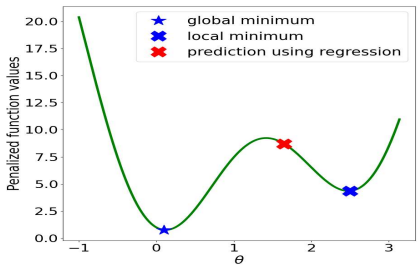


Fig. 4: When the training set has both local and global solutions, the predicted warm start using direct regression is marked as red, which is closer to the strict local minimum. A solver initialized with this warm start would converge to the strict local minimum.

global solutions, the predicted multiplier would be some point lying between the locally and globally optimal values. The predicted multiplier is plotted in Fig. 5a, where we adopt a linear cost function for $c(\cdot)$ in (7) and set the cost coefficient to be one, i.e., $c(x) = x$. Then we predict the solution of $\mathcal{L}_\mu$ from the predicted multiplier. For a given multiplier $\bar{\mu}$, the solution to $\mathcal{L}_\mu$ is found through the optimality condition of (7):

$$(c' + \bar{\mu})g\sin(\bar{\theta}) + (c' - \bar{\mu})b\cos(\bar{\theta}) = 0, \tag{8}$$

where c' is a shorthand for the derivative $c'(g - g\cos(\bar{\theta}) + b\sin(\bar{\theta}))$. By varying the multipliers, we can represent the mapping from the multipliers to the solutions of $\mathcal{L}_\mu$ as follows:

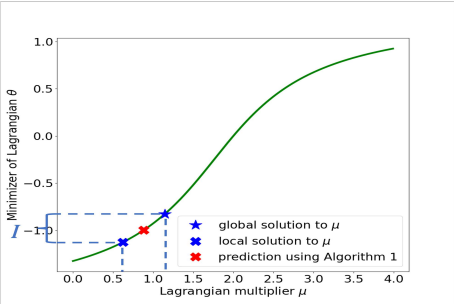
$$\bar{\theta} = \tan^{-1}\left(\frac{\bar{\mu} - c'}{\bar{\mu} + c'}b/g\right), \tag{9}$$

which is plotted in Fig. 5a. The set of solutions of $\mathcal{L}_\mu$ that is mapped from the multipliers varying between the locally and globally optimal values is denoted by set I. The predicted solution of $\mathcal{L}_\mu$ would lie in set I. We also plot set I in Fig. 5b. We can see that every point in set I is close to the global minimum of $\mathcal{L}_\rho$. More precisely, every point is in the basin of attraction of the global solution. This means if we use the predicted solution of $\mathcal{L}_\mu$ as a warm start, the solver would converge to the global minimum.

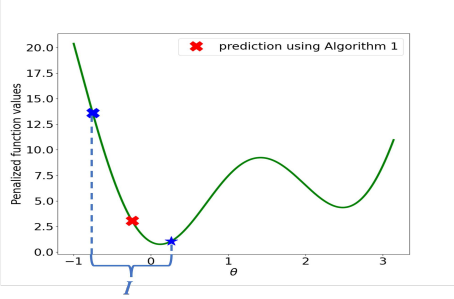
In the next section, we test Algorithm 1 on IEEE benchmark systems and show the intuition developed in this section is true for much larger and more complex problems.

V. Simulation Results

In this section, we demonstrate the simulation results of using Algorithm 1 to predict solutions to the ACOPF problem. We test our algorithm on IEEE networks with 22, 39, and 118 buses. The full specifications for these networks can be found in [11] and [23]. The popular solver IPOPT [24] is used to generate training samples for each network. For a comparison baseline, we use the method in [7], where a deep neural network is trained to learn the mapping from load to optimal generation values by minimizing the loss between the learned and ground-truth values. Then power flow equations are solved to recover and ensure the feasibility of the overall ACOPF solutions.



(a) The solutions of the partial Lagrangian as a function of the predicted multipliers. When the training data set is a mix of local and global solutions, the predicted multiplier and the associated solution of the partial Lagrangian are marked as red. The set I corresponds to the set of all possible solutions.



(b) The learned warm start point using Algorithm 1 is marked as red, which is close to the global minimum of the ACOPF problem. If this predicted warm start is used, the solver would converge to the global solution.

Fig. 5: Use Algorithm 1 to learn a warm start point for the ACOPF solver.

Our method can obtain globally optimal solutions even when the training data only contains local solutions and using the warm starts learned by our algorithm can speed up the computation time of solving ACOPF problems using IPOPT.

We use fully-connected neural networks with 2 hidden layers for both Algorithm 1 and the baseline method. For the baseline method, the activation function of the neural network is sigmoid for all layers. For Algorithm 1, we use ReLU as an activation function except for the output layer, where a linear activation function is used. All neural network models are implemented using the Tensorflow software library.

A. 22-bus Network

In the 22-bus network, there exist two solutions for a given load, where the cost of the local solution is 30% higher than that of the global solution. We generate the training data by varying the load around the nominal value. For each given load, we solve the ACOPF problem using IPOPT to obtain both solutions (this is done by using a number of random initial points). Then we construct 5 different training sets by adjusting the proportion of strictly local solutions in the data. There are 4000 training samples. We use 90% of them for training and 10% for testing.

The generation costs of the obtained solutions using both methods on different training sets are reported in Fig. 6, where the generation costs are represented proportionally to the globally optimal cost. In Fig. 6, as the proportion of strictly

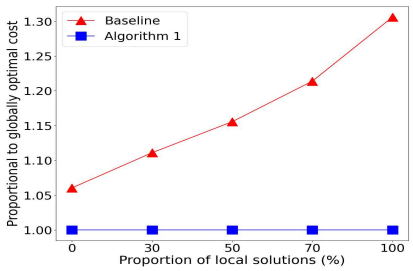


Fig. 6: Normalized generation costs of the obtained solutions using Algorithm 1 and baseline algorithm for the 22-bus network as the proportion of strictly local solutions in the training set increases. Algorithm 1 is not sensitive to the quality of training data and is able to obtain the global solution, where the performance of the baseline learning method degrades as the training data quality degrades.

local solutions in the training set increases, the predicted cost using the baseline method also increases and is larger than the globally optimal cost on every training set. In contrast, Algorithm 1 is able to obtain the global solution, even when the training set is comprised only of strictly local solutions. This implies that Algorithm 1 is not sensitive to the quality of the training set, and local solutions can also be useful. This observation carries over to larger networks.

B. 39-bus Network

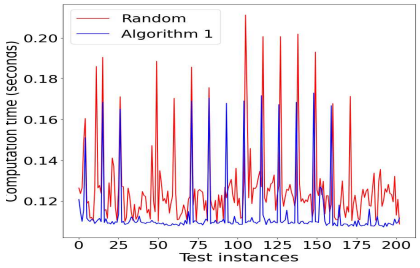


Fig. 7: Computation time of calling IPOPT to solve the ACOPF problem in 39-bus network with different initialization. The blue curve is the computation time when the warm starts learned by Algorithm 1 are used as initial points, which is lower than the random initialization (red curve) almost for every instance.

A key benefit of using machine learning for ACOPF is to speed up computation. Here, we take the IEEE 39-bus network and compare the solution speed of using Algorithm 1 to that of directly using IPOPT on random (Gaussian) initial points. We evaluate the computation time on Macbook Pro with Intel Core i5 8259U CPU @ 2.30GHz.

We call IPOPT with both initializations for 200 instances and report the computation time for each instance in Fig. 7. The computation time of using the learned warm starts given by Algorithm 1 is plotted as the blue line, which is much faster than the random initialization (red) almost for every instance. Note that the neural networks used in Algorithm 1 are feed-forward functions, and their evaluation time (sub-milliseconds) is negligible for the comparison in Fig. 7.

C. 118-bus Network

For the 118-bus network, there exist three solutions for a given load. The worst cost is 39% higher than the globally optimal cost. We generate the training data by varying the load around the nominal value. For each given load, we solve the ACOPF problem using IPOPT to obtain multiple solutions (this is done by using a number of random initializations). Then we construct 3 different training sets with different proportions of strict local solutions as shown in Fig 8. There are 1900 training samples and 90% used for training and 10% for testing. In Fig. 8, we compare the generation costs, which are normalized to the global optimal value, of the obtained solutions using Algorithm 1 to that predicted by the baseline method.

The predicted cost using the baseline method is larger than the globally optimal cost on every training set, and increases as the quality of the training data degrade (the proportion of local solutions increases). In contrast, Algorithm 1 is able to obtain the globally optimal cost regardless of the quality of the training data. Even when the training data only contains the solutions with the highest cost, the predicted cost using Algorithm 1 is globally optimal. This confirms with the intuition in Section IV that the predicted solutions of the partial Lagrangian would be close to the global minimum of the ACOPF problem, and hence could be good warm starts for the solver to reach the global solution.

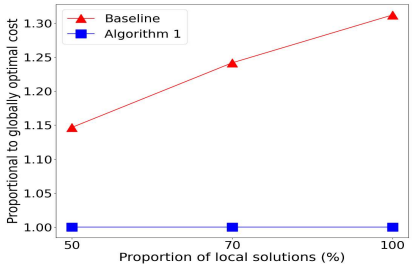


Fig. 8: Generation costs of the obtained solutions using Algorithm 1 and the baseline method on different training sets for the 118-bus network. All the generation costs are represented proportionally to the globally optimal cost. Algorithm 1 is able to obtain the global solution even when the training data only consists of local solutions.

VI. Conclusion

In this paper, we propose a partial Lagrangian-based learning approach to predict solutions of the ACOPF problem. First, we use a neural network to learn dual variables of the ACOPF problem. Then we use a second neural network to predict solutions of the partial Lagrangian from the predicted dual variables. Using the predicted solutions of the partial Lagrangian as warm starts, the ACOPF solver can reach more globally optimal solutions. We validate the effectiveness of our algorithm on standard 22-bus, 39-bus and 118-bus networks, and show our algorithm is able to obtain the globally optimal solutions and offer significant speedups.

References

[1] A. Castillo and R. P. O'Neill, "Computational performance of solution techniques applied to the acopf," *Federal Energy Regulatory Commission, Optimal Power Flow Paper*, vol. 5, 2013.

[2] X. Pan, T. Zhao, and M. Chen, "Deepopf: Deep neural network for dc optimal power flow," 2020.

[3] N. Guha, Z. Wang, M. Wytock, and A. Majumdar, "Machine learning for ac optimal power flow," 2019.

[4] 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*, 2020, pp. 5930–5934.

[5] Y. Zhou, B. Zhang, C. Xu, T. Lan, R. Diao, D. Shi, Z. Wang, and W.-J. Lee, "A data-driven method for fast ac optimal power flow solutions via deep reinforcement learning," *Journal of Modern Power Systems and Clean Energy*, vol. 8, no. 6, pp. 1128–1139, 2020.

[6] M. K. Singh, V. Kekatos, and G. B. Giannakis, "Learning to solve the ac-opf using sensitivity-informed deep neural networks," *ArXiv:2103.14779*, 2021.

[7] K. Baker, "Emulating ac opf solvers for obtaining sub-second feasible, near-optimal solutions," *ArXiv:2012.10031*, 2020.

[8] H. Lange, B. Chen, M. Berges, and S. Kar, "Learning to solve ac optimal power flow by differentiating through holomorphic embeddings," *arXiv:2012.09622*, 2020.

[9] A. Zamzam and K. Baker, "Learning optimal solutions for extremely fast ac optimal power flow," 2019.

[10] J. Momoh, R. Koessler, M. Bond, B. Stott, D. Sun, A. Papalexopoulos, and P. Ristanovic, "Challenges to optimal power flow," *IEEE Transactions on Power systems*, vol. 12, no. 1, pp. 444–455, 1997.

[11] W. A. Bukhsh, A. Grothey, K. I. McKinnon, and P. A. Trodden, "Local solutions of the optimal power flow problem," *IEEE Transactions on Power Systems*, vol. 28, no. 4, pp. 4780–4788, 2013.

[12] D. Wu, D. K. Molzahn, B. C. Lesieutre, and K. Dvijotham, "A deterministic method to identify multiple local extrema for the ac optimal power flow problem," *IEEE Transactions on Power Systems*, vol. 33, no. 1, pp. 654–668, 2017.

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

[14] D. P. Bertsekas, "Nonlinear programming," *Journal of the Operational Research Society*, vol. 48, no. 3, pp. 334–334, 1997.

[15] L. Zhang and B. Zhang, "An iterative approach to improving solution quality for ac optimal power flow problems," in *Proceedings of the ACM E-Energy*, 2022, p. 289–301.

[16] P. L. Donti, D. Rolnick, and J. Z. Kolter, "Dc3: A learning method for optimization with hard constraints," *arXiv:2104.12225*, 2021.

[17] W. Ma and J. S. Thorp, "An efficient algorithm to locate all the load flow solutions," *IEEE Transactions on Power Systems*, vol. 8, no. 3, pp. 1077–1083, 1993.

[18] J. A. Momoh, R. Adapa, and M. El-Hawary, "A review of selected optimal power flow literature to 1993. i. nonlinear and quadratic programming approaches," *IEEE transactions on power systems*, vol. 14, no. 1, pp. 96–104, 1999.

[19] B. Lesieutre and D. Wu, "An efficient method to locate all the load flow solutions-revisited," in *2015 53rd Annual Allerton Conference on Communication, Control, and Computing*, 2015, pp. 381–388.

[20] B. Lesieutre, J. Lindberg, A. Zachariah, and N. Boston, "On the distribution of real-valued solutions to the power flow equations," in *2019 57th Annual Allerton Conference on Communication, Control, and Computing*, 2019, pp. 165–170.

[21] M. B. Cain, R. P. O'neill, A. Castillo *et al.*, "History of optimal power flow and formulations," *FERC*, vol. 1, pp. 1–36, 2012.

[22] J. Mulvaney-Kemp, S. Fattahi, and J. Lavaei, "Load variation enables escaping poor solutions of time-varying optimal power flow," in *PESGM*, 2020.

[23] H. D. Nguyen and K. S. Turitsyn, "Appearance of multiple stable load flow solutions under power flow reversal conditions," in *2014 IEEE PES General Meeting— Conference & Exposition*. IEEE, 2014, pp. 1–5.

[24] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical programming*, vol. 106, no. 1, pp. 25–57, 2006.