A Directed Acyclic Graph Neural Network for AC Optimal Power Flow

Zhenping Guo[1], Kai Sun[1], *Senior Member*, *IEEE*, Byungkwon Park[2], *Member*, *IEEE*, Srdjan Simunovic[3], Wei Kang[4], *Fellow*, *IEEE*
[1]Department of EECS, University of Tennessee, Knoxville, TN, USA
[2]Department of Electrical Engineering, Soongsil University, Seoul, South Korea
[3]Computational Sciences and Engineering Division, Oak Ridge National Laboratory, Oak Ridge, TN, USA
[4]Department of Applied Mathematics, Naval Postgraduate School, Monterey, CA, USA
zguo19@vols.utk.edu, kaisun@utk.edu, bkpark@ssu.ac.kr, simunovics@ornl.gov, wkang@nps.edu

***Abstract*—AC optimal power flow (OPF) is of great significance for power system security, reliability, and economy. As an NP-hard problem, its solution can be time consuming by traditional optimization techniques. For more efficient AC OPF algorithms, a Direct Acyclic Graph Neural Network (DAG-NN) is proposed in this paper, which enables an explicit design of a neural network utilizing the intrinsic structural information of the problem to be solved. The approach first reformulates an iterative Newton-Raphson based AC OPF algorithm as a compositional function, accordingly constructs a DAG, and then designs the neural network by realizing its each node by a shallow neural network. The paper also analyzes errors of the DAG-NN. The proposed approach is tested on a modified PJM 5-bus system.**

***Index Terms*--AC optimal power flow, compositional function, directed acyclic graph, DAG, neural network.**

I. Introduction

As a vital reference for power system operation, AC optimal power flow (OPF) plays an important role in power system planning, economic dispatch and system control [1]-[3]. However, due to the nonconvexity of a nonlinear AC OPF model, its numerical solution can be time-consuming if many iterations are needed. To reduce the difficulty of AC OPF, relaxation approaches have been utilized by [4]-[6] to make the problem convex by, e.g., the second-order cone program, semi-definite programming, and the quadratic convex relaxation.

In recent years, learning-based tools have been widely used to solve power system problems such as load forecasting [7], vulnerability analysis [8], transient stability prediction [9]-[10], and voltage stability assessment [11]-[12]. These tools shift a majority of computational burdens, such as trainings based on big data, to an offline stage, so they can be highly efficient in the online stage for real-time applications. Many studies have applied machine learning (ML) techniques to solve the AC OPF. Some used ML to predict final AC OPF solutions directly [13], and others used ML to find a good initial solution, which can be further refined by a traditional numerical solver. For example, [14] adopted ML in predicting warm start points of the AC OPF to accelerate the computational speed of numerical methods, and [15] utilized ML to classify activations of constraints. However, the applications of ML approaches often ignore the intrinsic information of the structure of the AC OPF model. Thus, a question arises: can we utilize the mathematical structure of an AC OPF model to design the ML tool such as a neural network (NN)? Compact Directed Acyclic Graphs (DAGs) have been used in literature to help design the structure of a neural network [16]-[17]. From [17], the solution of any problem can be essentially formulated as a compositional function, which can be represented by a DAG and then be further approximated by an NN. Inspired by [17], this paper develops a Directed Acyclic Graph Neural Network (DAG-NN) to address the computational challenges of the AC OPF problem. Also, there is no theoretical way to determine the structure of a traditional NN, i.e., the number of layers and neurons. The DAG-NN provides a way to utilize the intrinsic information of the AC OPF structure to explicitly design the structure of the NN and thus have the potential to improve accuracy. In this work, we target at finding an AC OPF solution close to the true solution. If a high accuracy is desired, the solution can be used as a good initial guess and be refined by a numerical solver.

In the rest of the paper, Section II describes how to construct the proposed DAG-NN for AC OPF; Section III presents two case studies on a 2-bus toy system and a modified PJM 5-bus system. Conclusions and future work are summarized in Section IV.

This work was supported in part by UT-Battelle, LLC, through the U.S. Department of Energy under Contract DE-AC05-00OR22725.

II. Proposed DAG-NN for AC OPF

This section first introduces the basic concept of a DAG to represent a compositional function, and then describes how to use a DAG to construct an NN to approximate the compositional function, and finally presents how to formulate the AC OPF problem as a compositional function and build a DAG-NN for AC OPF.

A. Introduction of DAGs

The DAGs in this paper are composed of a finite number of nodes and directed arrows, where each node represents a simple function, and directed arrows determine the interconnections of those simple functions. A DAG does not have directed cycles.

Considering a compositional function below:

$$f(x_1, x_2, x_3) = x_1x_2 - x_2x_3 + \cos x_3 \tag{1}$$

Its DAG is shown in Fig. 1, where the nodes represent simple functions of the compositional function. In the graph, the nodes are marked with different colors that represent linear and nonlinear functions. The DAG of this compositional function can be used to construct an NN by replacing its each node by a shallow NN.

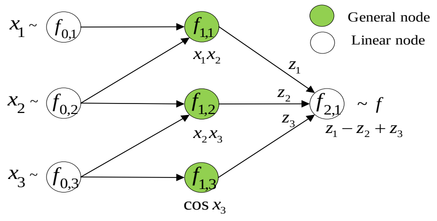


Figure 1. A DAG of the compositional function f

In general, the complexity of solving non-convex OPF problems grows exponentially with the dimensionality. However, by taking advantage of the compositional features of DAGs, it is proved in [17] that the complexity of solving the problem using NN has a complexity that grows at a polynomial rate. A compositional function $\mathbf{f}$ represented by a DAG can be approximated by replacing each node of the DAG with a shallow NN, and then the DAG is approximated by a NN $\tilde{\mathbf{f}}$.

The error of the NN satisfies (2) which is proved in [17].

$$\left\|\tilde{\mathbf{f}}(x)-\mathbf{f}(x)\right\|_p \le \sum_j L_j^{\mathbf{f}} \varepsilon_j \tag{2}$$

where ε_j is the local error caused by the j^{th} node approximation with the shallow NN; $L_j^{\mathbf{f}}$ is the Lipschitz constant associated with the j^{th} node; and the left-hand side is the error defined by p^{th} norm [17].

B. Formulation of AC OPF

The AC OPF problem is formulated as below:

$$\min\ F=\sum_{i=1}^{n_g} c_{2i}PG_i^2+c_{1i}PG_i+c_{0i} \tag{3a}$$

$$\left.\begin{aligned} s.t\ \Delta P_i &= PG_i - PD_i - V_i\sum_{j=1}^{n_b} V_j Y_{ij}\cos(\delta_i-\delta_j-\theta_{ij}) \\ \Delta Q_i &= QG_i - QD_i - V_i\sum_{j=1}^{n_b} V_j Y_{ij}\sin(\delta_i-\delta_j-\theta_{ij}) \end{aligned}\right\} (i=1,2...n_b) \tag{3b}$$

$$V_{i,min} \le V_i \le V_{i,max}\quad (i=1,2...n_b) \tag{3c}$$

$$\left.\begin{aligned} PG_{i,min} &\le PG_i \le PG_{i,max} \\ QG_{i,min} &\le QG_i \le QG_{i,max} \end{aligned}\right\}(i=1,2...n_g) \tag{3d}$$

$$|S_l| \le |S_l|_{max}\ (l=1,2...n_l) \tag{3e}$$

where c_{0i}, c_{1i}, c_{2i} are coefficients of the cost function; PG_i and QG_i represent the active and reactive power outputs of generator i, respectively; Y_{ij} and θ_{ij} represent the magnitude and angle of the admittance, respectively; V_i and δ_i represent the magnitude and angle of the voltage, respectively; PD_i and QD_i are the active and reactive power demands, respectively; $|S_l|$ is the line flow magnitude; n_g, n_b and n_l represent the number of generators, buses and lines, respectively.

The AC OPF problem in (3) can be converted into an unconstrained problem by applying Lagrange multipliers to power flow constraints given in (3b). Then the cost function given in (3a) becomes the Lagrange function:

$$\mathcal{L}=\sum_{i=1}^{n_g} c_{2i}PG_i^2+c_{1i}PG_i+c_{oi}-\lambda_{pi}\Delta P_i-\lambda_{qi}\Delta Q_i \tag{4}$$

Thus, the original problem is transformed to the problem described by the equation below:

$$\nabla\mathcal{L}=0 \tag{5}$$

The solution of (5) may not be unique, and one may find a local minimum because of the non-convexity. Thus, the DAG-NN is applied in this paper to reduce the complexity of this problem.

C. The DAG-NN for AC OPF

Solving the AC OPF problem can be formulated as a compositional function, and the compositional function can be represented by a DAG. Then, the DAG is utilized to design the DAG-NN. In this way, the DAG-NN takes advantage of the mathematical structure of the AC OPF and has the potential to improve accuracy compared with a traditional NN designed without any structural information of the problem. To formulate the compositional function of AC OPF, the Newton-Raphson method is applied to solve (5) by iterations which each calculates:

$$\mathbf{y}=[PG,QG,\delta,V,\lambda] \tag{6a}$$

$$\mathbf{y}^{k+1}=\mathbf{y}^k+\Delta\mathbf{y}^k=\mathbf{y}^k+\mathbf{h}(\mathbf{y}^k) \tag{6b}$$

$$N(\mathbf{y})=\mathbf{y}+\mathbf{h}(\mathbf{y}) \tag{6c}$$

$$\mathbf{h}(\mathbf{y})=-\left[\nabla^2\mathcal{L}\right]^{-1}\times\nabla\mathcal{L} \tag{6d}$$

where $\mathbf{y}$ is the current variable vector; $\mathbf{h}$ is the correction, i.e., $\Delta\mathbf{y}$ for each iteration; k denotes the current iteration number, k+1 denotes the next iteration; $N(\mathbf{y})$ includes the updated variables.

From the definition of DAGs in subsection II-*A*, one iteration of the Newton-Raphson method can be represented by a DAG, named 1-DAG, shown below in Fig. 2. In addition, a *K*-DAG is defined as the series connection of *K* repetitions of the same 1-DAG, which approximates the computation by *K* iterations of the Newton-Raphson method. Also, $\mathbf{h}(\mathbf{y})$ represents $\Delta\mathbf{y}$ in each iteration, which is also a compositional function and will be illustrated in detail in subsection II-*E*.

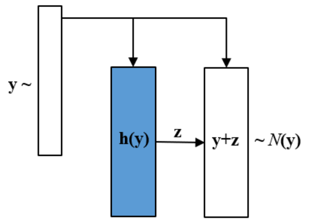


Figure 2. 1-DAG of one Newton-Raphson iteration

As illustrated above, solving the AC OPF by the Newton-Raphson method is equal with solving the same compositional functions repeatedly. Therefore, the compositional function in one iteration can be represented by a DAG, and each node of the DAG which represents a simple function can be estimated by a shallow NN. Thus, a DAG-NN for AC OPF is constructed by replacing each node of the *K*-DAG with a shallow NN. Moreover, according to the mathematical forms of the AC OPF shown in (3)-(6), new activation functions including the sinusoidal function ($\sin x$) and the quadratic function (x^2) are applied to the DAG-NN.

D. Error analysis on the DAG-NN

Suppose that the number of iterations for the Newton-Raphson method is *K*, and then the optimal solution can be expressed as follows:

$$\phi(\mathbf{y})\approx\overbrace{N(\cdot)\circ\cdots N(\cdot)\circ N}^{K}(\mathbf{y})=\left(N(\cdot)\right)^K(\mathbf{y}) \tag{7}$$

According to subsection II-*A*, there exists an NN approximating **h** satisfying:

$$\left\| \mathbf{h}(\mathbf{y}) - \mathbf{h}^{NN}(\mathbf{y}) \right\|_p \le \sum_j L_j^{\mathbf{h}} \varepsilon_j \tag{8}$$

where ε_j is the local error caused by the j^{th} node approximation with the shallow NN, and $L_j^{\mathbf{h}}$ is the Lipschitz constant associated with the j^{th} node.

Let N^{NN} (**y**) be the NN obtained by substituting $\mathbf{h}^{NN}(\mathbf{y})$ for $\mathbf{h}(\mathbf{y})$ in (6c), and then we have:

$$\begin{aligned} \left\| N(\mathbf{y}) - N^{NN}(\mathbf{y}) \right\|_p &= \left\| \mathbf{y} + \mathbf{h}(\mathbf{y}) - (\mathbf{y} + \mathbf{h}^{NN}(\mathbf{y})) \right\|_p \\ &= \left\| \mathbf{h}(\mathbf{y}) - \mathbf{h}^{NN}(\mathbf{y}) \right\|_p \le \sum_j L_j^{\mathbf{h}} \varepsilon_j \end{aligned} \tag{9}$$

Define: $\phi^{NN} = \left(N^{NN}(\cdot) \right)^K$. Let the Lipschitz constant of N(**y**) be *L*. Applying the Proposition 3.10 in [17], we can get

$$\left\| \left(N(\cdot) \right)^K (\mathbf{y}) - \phi^{NN}(\mathbf{y}) \right\|_p \le \frac{L^K - 1}{L - 1} \sum_j L_j^{\mathbf{h}} \varepsilon_j \tag{10}$$

Suppose that the error tolerance of the Newton-Raphson method is e_N. Namely

$$\left\| \phi(\mathbf{y}) - \left(N(\cdot) \right)^K (\mathbf{y}) \right\|_p \le e_N \tag{11}$$

Finally, the error of the NN for AC OPF can be obtained from (10) and (11) in the following way:

$$\begin{aligned} \left\| \phi(\mathbf{y}) - \phi^{NN}(\mathbf{y}) \right\|_p &= \left\| \phi(\mathbf{y}) - \left(N(\cdot) \right)^K (\mathbf{y}) + \left(N(\cdot) \right)^K (\mathbf{y}) - \phi^{NN}(\mathbf{y}) \right\|_p \\ &\le \left\| \left(N(\cdot) \right)^K (\mathbf{y}) - \phi^{NN}(\mathbf{y}) \right\|_p + \left\| \phi(\mathbf{y}) - \left(N(\cdot) \right)^K (\mathbf{y}) \right\|_p \\ &\le \frac{L^K - 1}{L - 1} \sum_j L_j^{\mathbf{h}} \varepsilon_j + e_N \end{aligned} \tag{12}$$

E. *The prediction of DAG structure for AC OPF*

The DAG of the AC OPF is used to construct the structure of an NN approximating the solution of AC OPF. However, to construct the DAG-NN for the AC OPF, the explicit expression of function **h** is required, which cannot be obtained for most high-dimensional systems due to the complexity of calculating matrix inverse. To simplify the DAG, the chord method [18] is applied replacing a standard Newton-Raphson method, and conducts (13) for each iteration, where *B* is a constant matrix approximating the inverse of matrix $\nabla^2 \mathcal{L}$.

$$N(\mathbf{y}) = \mathbf{y} + \mathbf{h}(\mathbf{y}) \tag{13a}$$

$$\mathbf{h}(\mathbf{y}) = -B \times \nabla \mathcal{L} \tag{13b}$$

Thus, the DAG structure of function **h**(**y**) becomes a linear combination of all the basic elements of $\nabla \mathcal{L}$, which can be easily obtained and applied to large systems. According to subsection II-*B*, the basic elements of $\nabla \mathcal{L}$ consist of only three different operations: (a) linear operations; (b) sinusoid operation, which can be realized by neurons with a sinusoid activation function; (c) multiplication operation. Because $xy = [(x+y)^2 - (x-y)^2]/4$, the multiplication operation can be realized by quadratic and linear neurons.

Fig. 3 shows a general DAG of function **h**(**y**) constructed from the chord method. Here, **y** is an *n*-dimensional vector, and $\mathbf{h}(\mathbf{y}) = [h_1, h_2, \ldots, h_n]^T$. Part (a) includes pure linear nodes, which are used to conduct the linear operations for the previous layer; part (b) includes sinusoid neurons; part (c) and part (d) together act as multiplication operations, and the same for the rest such as (e) and (f), (g) and (h). Here, we just show a typical structure with 9 layers. However, the specific layers can be adjusted for different cases.

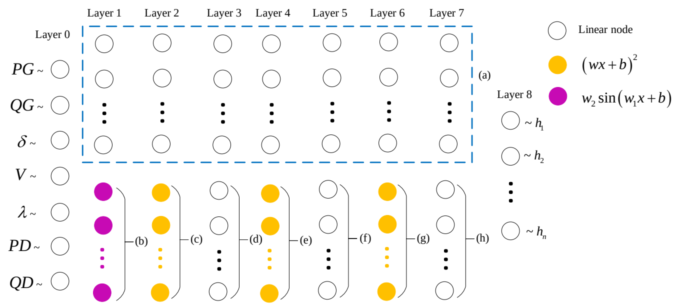


Figure 3. A DAG of function **h**(**y**) based on the chord method

III. Case Studies

This section first illustrates the idea of the proposed DAG-NN on a 2-bus toy system. Next, the DAG-NN is tested on a modified PJM 5-bus system. All simulations are conducted on a desktop computer with Intel core i7 CPU and 16GB RAM.

A. *2-bus system*

A 2-bus system is shown in Fig. 4. It has two generators G1 and G2, and one load PL₂. The costs of the two generators are $C_1 : 0.01P_1^2 + 2P_1$ \$/hr and $C_2 : 0.03P_2^2$ \$/hr, respectively.

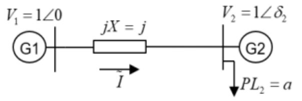


Figure 4. A 2-bus system

Setting the based power as 100 MVA, the AC OPF problem can be formulated as below:

$$\begin{aligned} &\min\ C = 0.01 \times (100PG_1)^2 + 2 \times (100PG_1) + 0.03 \times (100PG_2)^2 \\ &s.t.\ PG_1 + PG_2 = PL_2 = a \\ &\quad PG_1 = V_1V_2 \sin(\delta_1 - \delta_2)/X = 1 \times 1 \times \sin(0 - \delta_2)/1 = -\sin\delta_2 \end{aligned} \tag{14}$$

Equation (14) can be simplified as:

$$\begin{aligned} &\min\ C = 100(PG_1^2 + 2PG_1 + 3PG_2^2) \\ &s.t.\ PG_1 + PG_2 = PL_2 = a \\ &\quad PG_1 = -\sin\delta_2 \end{aligned} \tag{15}$$

The problem defined in (15) is equivalent to the problem formulated below:

$$\begin{aligned} &\min\ \tilde{C} = PG_1^2 + 2PG_1 + 3PG_2^2 \\ &s.t.\ PG_1 + PG_2 = PL_2 = a \\ &\quad PG_1 = -\sin\delta_2 \end{aligned} \tag{16}$$

Apply Lagrange multipliers to (16). Here is:

$$\mathcal{L} = (-\sin\delta_2)^2 - 2\sin\delta_2 + 3PG_2^2 - \lambda(-\sin\delta_2 + PG_2 - a) \tag{17}$$

For simplicity, changing the notations of two variables following *u*=*PG*₂ and x=δ₂, (17) becomes (18):

$$\mathcal{L} = (-\sin x)^2 - 2\sin x + 3u^2 - \lambda(-\sin x + u - a) \tag{18}$$

The local minimum of (18) is obtained by solving (19) below with zero gradient of the objective function:

$$\nabla \mathcal{L} = \begin{bmatrix} \frac{\partial C^*}{\partial \lambda} & \frac{\partial C^*}{\partial x} & \frac{\partial C^*}{\partial u} \end{bmatrix}^T = \begin{bmatrix} a - u + \sin x \\ (-2 + \lambda + 2\sin x)\cos x \\ 6u - \lambda \end{bmatrix} = 0 \tag{19}$$

For the 2-bus system, we use both the standard Newton-Raphson method and chord method to construct the DAG-NN.

1) *Error analysis of DAG-NN on the 2-bus system*

By substituting (19) to (6d), $\mathbf{h}(\mathbf{y})$ and $N(\mathbf{y})$ of the standard Newton-Raphson method can be obtained, and a DAG of $\mathbf{h}(\mathbf{y})$ for the 2-bus system is shown in Fig. 5. In the DAG, the nodes representing simple functions are approximated by shallow NNs, each of which is trained separately. A DAG-NN named "1-DAG-NN" that approximates one Newton-Raphson iteration is generated by substituting all these shallow NNs into Fig. 2. Then another DAG-NN called "4-DAG-NN" is constructed by integrating four 1-DAG-NNs in a series way, which corresponds to four iterations in terms of the Newton-Raphson method. The 4-DAG-NN is used for solving the AC OPF of the 2-bus system here.

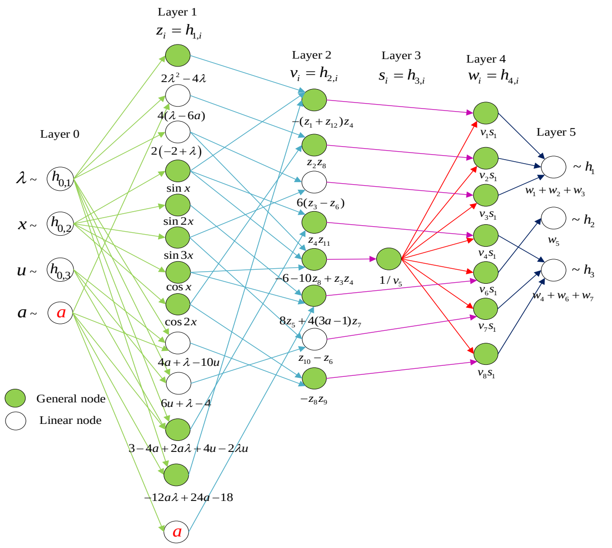


Figure 5. A DAG of function $\mathbf{h}(\mathbf{y})$ based on the Newton-Raphson method

A dataset for the NN training and testing is generated by solving the optimal power flow with the Newton-Raphson method under different loading conditions within a variation of 30%. The error tolerance of the Newton-Raphson method is set as 10^{-10}, and solutions that can converge within 4 iterations are picked to make up the dataset.

In this case, the $l\infty$-norm is used to measure the error. ε_j can be obtained by calculating the $l\infty$-norm of the test data error for the j^{th} shallow NN; The Lipschitz constant L of $N(\mathbf{y})$ is determined by the expression of $N(\mathbf{y})$; The Lipschitz constant associated with the j^{th} node $L_j^{\mathbf{h}}$ defined in [17] can be calculated according to the DAG shown in Fig. 5. According to (12), the error upper bound of the 4-DAG-NN is:

$$\begin{aligned}\left\|\phi(\mathbf{y}) - \phi^{NN}(\mathbf{y})\right\|_\infty &\le \frac{L^K - 1}{L-1}\sum_j L_j^{\mathbf{h}}\varepsilon_j + e_N \\ &= \frac{71.4139^4 - 1}{71.4139 - 1} \times 4.372\times10^{-13} + 1\times10^{-10} \\ &= 1.6\times10^{-7}\end{aligned} \tag{20}$$

After training the 4-DAG-NN with the training dataset, the AC OPF is predicted. All the predicted results are found to satisfy the constraints and are feasible, and then the error is calculated by comparing the results with true values. Finally, the $l\infty$-norm of test data error for the 4-DAG-NN is obtained as 1.02E-09, which is smaller than the error upper bound calculated above. This validates that the DAG can be utilized to design NNs, and the error upper bound can be calculated according to (12).

2) *DAG-NN performance of the 2-bus system*

To reduce the complexity of the DAG, a DAG-NN based on the chord method instead of the standard Newton-Raphson method is then constructed. In this case, three DAG-NNs are constructed based on the chord method with different repetitions of the DAG structure. Three traditional NNs with the same number of neurons are also constructed for comparison. Case 1 and case 4 have the same number of neurons, and the same for case 2 and case 5, as well as case 3 and case 6. For this 2-bus system, one DAG structure consists of 22 neurons, so a K-DAG has $22\times K$ neurons. To avoid influence of random factors, 100 NN models are trained with different initial parameters for each case. The errors of the top 1, top 10, and top 50 results using different NNs are presented in Table I. Table II shows the time costs for the 2-bus system. As shown in Table I and Table II, all DAG-NNs have smaller mean absolute errors and time costs than traditional NNs with the same NN size. This corroborates the good performance of the DAG-NN.

TABLE I. MEAN ABSOLUTE ERRORS OF THE COST FUNCTION

Case \ Rank		Top 1	Top 10	Top 50
DAG-NNs	Case 1: 2-DAG-NN	1.08E-07	1.47E-06	2.18E-06
	Case 2: 3-DAG-NN	1.01E-08	9.20E-08	1.48E-06
	Case 3: 4-DAG-NN	7.59E-09	3.23E-08	3.21E-07
Traditional NNs	Case 4: 2 hidden layers	2.63E-06	5.58E-06	2.74E-05
	Case 5: 3 hidden layers	2.70E-06	3.45E-06	6.87E-06
	Case 6: 4 hidden layers	1.33E-06	2.50E-06	5.31E-06

TABLE II. TIME COSTS FOR ONE TEST DATA OF THE 2-BUS SYSTEM

Case		Time cost (ms)
DAG-NNs	Case 1: 2-DAG-NN	0.96
	Case 2: 3-DAG-NN	1.89
	Case 3: 4-DAG-NN	2.02
Traditional NNs	Case 4: 2 hidden layers	1.02
	Case 5: 3 hidden layers	2.00
	Case 6: 4 hidden layers	2.99

B. *Case Study on a modified PJM 5-bus system*

A modified PJM 5-bus system is shown in Fig. 6.

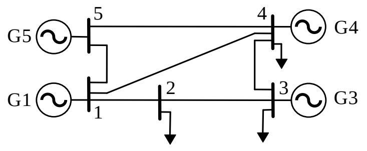
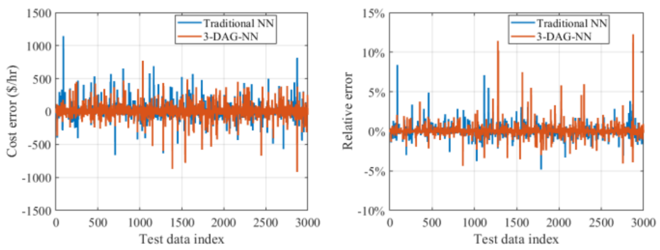


Figure 6. A modified PJM 5-bus system

A public dataset for learning AC OPF developed by NREL using a toolbox named 'Opf-learn' is used here [19]. Fig. 7 shows the results of a 3-DAG-NN and a traditional NN of the same size. In Fig. 7, errors of both the DAG-NN and the traditional NN are due to violations of power flow equations or inequality constraints. Therefore, a further correction step is required to improve the accuracy and feasibility. As a common practice, the predicted results provided by NNs are chosen as warm start points to resolve the AC OPF problem by the MATPOWER Interior Point Solver (MIPS). Starting from the warm start points obtained by DC OPF, the DAG-NN and the traditional NN, all the 3000 testing cases can converge to accurate results, whose relative errors of the costs are within [-0.0005%, 0.0005%]. Therefore, the DAG-NN can provide a good initial guess for the AC OPF. Although DC OPF can also provide a good initial point for the AC OPF, it is not as efficient as the proposed NN which can be observed in Table III. Because the DAG-NN is not a fully connected network, it has a smaller number of parameters compared with the traditional NN with the same number of neurons. It is expected to be more efficient than the traditional one, but the result shown in Table III is contrary to the expectation. One possible reason is that although the DAG-NN accelerates due to a smaller number of parameters, the customized construction of the DAG-NN is not as efficient as the traditional one which can be easily constructed from the package of PyTorch. From the case study, the DAG-NN is very competitive with the traditional NN. Moreover, the DAG-NN is promising in reducing the number of neurons thanks to the utilization of structural information of the problem. Its potential on larger cases will be tested in our future work.



(a) Error of cost function (b) Relative error of cost function

Figure 7. The error of the cost function for different NNs

TABLE III. TIME COSTS FOR ONE TEST DATA OF THE 5-BUS SYSTEM

Case	Time cost (ms)
DAG-NN	47.07
Traditional NN	45.34
DC OPF	61.28

IV. Conclusion

This paper formulates the process of solving AC OPF by the Newton-Raphson method as a compositional function, and the compositional function is then represented by a DAG, which is used to design and develop a NN. The error of the proposed DAG-NN was analyzed. Also, the DAG-NN was tested on a 2-bus system and a modified PJM 5-bus system. Case study results validate that the proposed DAG-NN has roughly equal or better performance than the traditional NN. Therefore, it is promising to include the DAG structure to design NNs. Furthermore, this approach also has good potential for other problems that can be formulated as a compositional function, not limited to the AC OPF discussed in this paper. The future work will focus on using other approaches instead of the chord method to achieve a more accurate DAG.

Acknowledgment

This manuscript has been authored by UT-Battelle, LLC under Contract No. DE-AC05-00OR22725 with the U.S. Department of Energy. The United States Government retains and the publisher, by accepting the article for publication, acknowledges that the United States Government retains a nonexclusive, paidup, irrevocable, world-wide license to publish or reproduce the published form of this manuscript, or allow others to do so, for United States Government purposes. The Department of Energy will provide public access to these results of federally sponsored research in accordance with the DOE Public Access Plan.

References

[1] S. S. Torbaghan, Gowri Suryanarayana, *et al*., “Optimal flexibility dispatch problem using second-order cone relaxation of AC power flows,” *IEEE Trans. Power Syst.*, vol. 35, no. 1, pp. 98-108, Jan. 2020.

[2] M. de Jong, G. Papaefthymiou, *et al*., “A framework for incorporation of infeed uncertainty in power system risk-based security assessment,” *IEEE Trans. Power Syst.*, vol. 33, no. 1, pp. 613-621, Jan. 2018.

[3] V. Rostampour, O. t. Haar, T. Keviczky, “Distributed stochastic reserve scheduling in ac power systems with uncertain generation,” *IEEE Trans. Power Syst.*, vol. 34, no. 2, pp. 1005-1020, March 2019.

[4] R. A. Jabr, “Radial distribution load flow using conic programming,” *IEEE Trans. Power Syst.*, vol. 21, no. 3, pp. 1458-1459, Aug. 2006.

[5] J. Lavaei, S. H. Low, “Zero duality gap in optimal power flow problem,” *IEEE Trans. Power Syst.*, vol. 27, no. 1, pp. 92-107, Feb. 2012.

[6] C. Coffrin, H. L. Hijazi, 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, July 2016.

[7] T. Senjyu, H. Takara, K. Uezato, T. Funabashi, “One-hour-ahead load forecasting using neural network,” *IEEE Trans. Power Syst.*, vol. 17, no. 1, pp. 113-118, Feb. 2002.

[8] Z. Zhang, R. Yao, S. Huang, Y. Chen, S. Mei, K. Sun, “An online search method for representative risky fault chains based on reinforcement learning and knowledge transfer,” *IEEE Trans. Power Syst.*, vol.35, no.3, pp.1856-1867, May 2020.

[9] A. Sepehr, O. Gomis-Bellmunt, E. Pouresmaeil, “Employing machine learning for enhancing transient stability of power synchronization control during fault conditions in weak grids,” *IEEE Trans. Smart Grid*, vol.13, no.3, pp.2121-2131, May 2022.

[10] M. C. Passaro, A. P. A. da Silva, A. C. S. Lima, “Preventive control stability via neural network sensitivity,” *IEEE Trans. Power Syst.*, vol. 29, no. 6, pp. 2846-2853, Nov. 2014.

[11] R. R Hossain, Q. Huang, R. Huang, “Graph convolutional network based topology embedded deep reinforcement learning for voltage stability control,” *IEEE Trans. Power Syst.*, vol. 36, no. 5, pp. 4848-4851, 2021.

[12] H. Hagmar, L. Tong, R. Eriksson, L. A. Tuan, “Voltage instability prediction using a deep recurrent neural network,” *IEEE Trans. Power Syst.*, vol. 36, no. 1, pp. 17-27, 2020.

[13] R. Canyasse, G. Dalal, S. Mannor, “Supervised learning for optimal power flow as a real-time proxy,” *2017 IEEE ISGT*, pp. 1-5, 2017.

[14] K. Baker, “Learning warm-start points for ac optimal power flow,” *2019 IEEE 29th MLSP*, pp. 1-6, 2019.

[15] X. Lei, Z. Yang, J. Yu, J. Zhao, Q. Gao, H. Yu, “Data-driven optimal power flow: A physics-informed machine learning approach,” *IEEE Trans. Power Syst.*, vol. 36, no. 1, pp. 346-354, Jan. 2021.

[16] C. Chiu, J. Zhan, An evolutionary approach to compact DAG neural network optimization, *IEEE Access*, vol. 7, pp. 178331-178341, Nov. 2019.

[17] W. Kang, Q. Gong, “Feedforward neural networks and compositional functions with applications to dynamical systems,” *SIAM Journal on Control and Optimization*, vol. 60, no. 2, pp. 786-813, Jan. 2021.

[18] V. S. Ryaben’kii, S. V. Tsynkov, “A theoretical introduction to numerical analysis,” *Chapman and Hall/CRC*, Nov. 2006.

[19] T. Joswig-Jones, K. Baker, A. S. Zamzam, “Opf-learn: An open-source framework for creating representative ac optimal power flow datasets,” *2022 IEEE ISGT*, pp. 1-5, April 2022.