Warm-Starting AC Optimal Power Flow with Graph Neural Networks

Frederik Diehl
fortiss
Research Institute of the Free State of Bavaria
associated with Technical University of Munich
Munich, Germany
diehl@fortiss.org

Abstract

Efficient control of transmission power grids is important both for efficiently managing generators and to prolong longevity of components. However, solving the optimization problem is NP-hard and linear approximations are necessary. The deployment of machine learning methods is hampered by the need to guarantee correctness. On a synthetic power grid modelling Texas, we show that a Graph Neural Networks (GNNs) produces a solution four orders of magnitude faster or can warm-start the optimizer for 3.75x faster convergence. This allows us to dispense with linear approximation, leads to more efficient generator dispatch, and can potentially save hundreds of megatons of $\mathrm{CO_2}$-equivalent.

1 Introduction

With a share of 25%, electricity and heat production make up a significant component of the yearly emissions of roughly $50\,\mathrm{GtCO_2}$-eq. (IPCC 2014, p. 46f). Accordingly, any increase in efficiency has the potential of a large impact. At the same time, energy supply is an extremely critical and sensitive system, in which blackouts and brownouts are unacceptable. Any new technology deployed here needs to guarantee correctness.

Day-to-day operating of an electrical grid requires scheduling of generator outputs. A Transmission System Operator (TSO) needs to optimize the purchase of power from different generators, each of which has different and potentially nonlinear costs and CO_2 emissions per produced MJ. Minimizing cost is known as Optimal Power Flow (OPF), and optimizing it exactly (referred to as ACOPF) has been proven to be NP-hard (Bienstock and Verma 2015). In the future, the problem size is bound to increase even more with the proliferation of small renewable generators.

Great progress has been achieved in the last decades (refer to Cain et al. (2012) for an overview). Yet, particularly in day-to-day operations which require solving OPF within a minute every five minutes, TSOs are forced to rely on linear approximations known as DCOPF. Solutions produced by these approximations are inefficient and therefore waste power and overproduce hundreds of megatons of $\mathrm{CO_2}$-equivalent per year.

We propose to use machine learning to produce a solution to the OPF problem. Knowing that such a solution will not necessarily be optimal or even legal (i.e. physically implementable), we can then use it to warm-start an ACOPF solver. Combining both approaches, we gain the best of both worlds: A significantly faster execution time and guaranteed legality.

This has previously been proposed by Guha et al. (2019). However, they used a feed-forward network and therefore have issues with both scalability and changing network geometry. They also did not integrate their results in an actual ACOPF optimizer.

2 Approach and Experiments

OPF is a complex problem governed by a great number of non-linear equations and constraints. We refer the reader to Frank and Rebennack (2016) for details on the problem formulation.

For the following, we approximate the problem as a graph: There are physical busses (each of which might contain loads, generators, and/or shunts) connected by branches. Each component is characterized by both physical parameters (for example a branch's resistance) and constraints (for example a generator's maximum power output).

By modelling the problem as a graph, we can apply GNNs to it. This approach allows us independence of the actual net topology, is data-efficient, and is suited to the sparse connectivity of a power network.

We rely on PowerModels.jl (Coffrin et al. 2018), and use IPOPT (Wächter and Biegler 2006) as an OPF solver. We implement the model using pytorch (Paszke et al. 2017) and the pytorch-geometric package (Fey and Lenssen 2019).

2.1 Model

In most applications and benchmarks, GNNs expect to have a set of homogeneously-typed nodes and edges connecting these. While we normally could model a bus' generator as being part of that bus, several generators can be linked to one bus. We therefore treat busses as connecting to sets of generators, loads, and shunts. Following Zaheer et al. (2017), we compute a representation of each of these components, then concatenate their summed features to the bus:

$$c^h = f_c^n\left(c^i\right), \quad o^0 = f_o^n\left(b^i \| \sum_{s \in N} s^h \| \sum_{g \in N} g^h \| \sum_{l \in N} l^h\right), \tag{1}$$

where c is one of s, b, g, and l (the shunt, bus, generator, and load features), ^{i} are the input features, ^{h} the encoded representation, ^{0} the 0th layer output, f are a set of learned functions (parameterized neural networks), and $\|$ is concatenation. N refers to the corresponding neighbours of each node. Afterwards, we compute edge features e and node features for layer n as

$$e^n = f_e^n\left(b^n \| e^{n-1} \| o^n\right), \qquad b^n = f^b\left(o^n \| \sum_{o,e \in N} f_{be}^n\left(o^n \| e^n\right)\right). \tag{2}$$

For prediction, we concatenate the encoded features and the corresponding node features to the generators. Afterwards, we apply separate feed-forward models to the features of each generator, bus, and branch. This architecture enables the output of both node, edge, and node-associated predictions. This is similar to the graph networks conceptualized by Battaglia et al. (2018).

2.2 Dataset

Due to security concerns, accurate real-world data on power networks is sparse, out-dated, or inaccurate (Birchfield et al. 2018). While a variety of example scenarios have been published, most of these are used for software correctness tests and therefore both unrealistic and orders of magnitude smaller than real power networks. Birchfield et al. (2018) introduced a methodology to construct synthetic power networks based on known powerplants, census information, and geographic information. They produced several example datasets, of which we use CASE_ACTIVSG200 and CASE_ACTIVSG2000. These are synthetic representations of the central part of Illinois (containing 200 busses and 230 and 115 kV networks) and of Texas (with 2000 busses and 500, 230, 161, and 115 kV networks). Maps of both can be found in the Appendix.

Both datasets have synthetic hourly load distributions available for one year, and we split these into train, validation, and test dataset according to the days. In the TEXAS dataset, total load drops below minimum generator output for about half the datapoints. Since our OPF solver does not support disabling generators, we ignore generator minimum production for TEXAS.

Method	Mean Time	95% Time
Model	0.03 s	0.04 s
DC	0.13 s	0.18 s
Model $\rightarrow$ AC	0.75 s	1.96 s
DC $\rightarrow$ AC	0.98 s	2.58 s
AC	1.03 s	3.58 s

(a) Performance on ILLINOIS.

Method	Mean Time	95% Time	Legal (of 2208)
Model	0.2 s	0.3 s	2208
DC	3.2 s	3.6 s	2062
Model $\rightarrow$ AC	243.9 s	621.2 s	2208
DC $\rightarrow$ AC	849.7 s	919.4 s	2208
AC	862.6 s	1665.9 s	2208

(b) Performance on TEXAS.

Table 1: Performance comparison on both datasets. On ILLINOIS, all methods find legal solutions. DC and AC are the corresponding power flow models, while $\rightarrow$ depicts warm-starting the ACOPF.

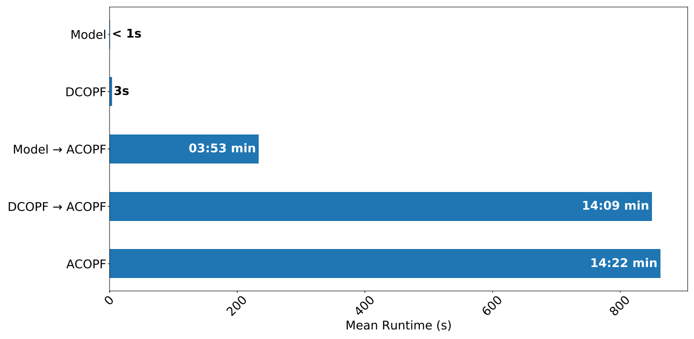

Figure 1: Runtime comparison on TEXAS.

3 Results and discussion

Table 1 and Fig. 1 show our result. We report both mean and $95\,\%$ quantile runtime. The latter is important for deployed systems. For evaluation, we do not run the model on a GPU.

ILLINOIS On the ILLINOIS dataset, consisting of 200 nodes, all models find legal solutions for all samples in our test set. The GNN runs 4.5x faster than DCOPF, and 36x faster than ACOPF. Even for such a small power grid, the combination of model and ACOPF saves $25\,\%$ runtime compared to the pure ACOPF, and is faster than using DCOPF to warm-start the optimization.

TEXAS On the more realistic TEXAS dataset, the pure model again improves on DCOPF by a factor of 10. Interestingly, it is able to solve all observed cases, while being faster than ACOPF by four orders of magnitude. Yet, its blackbox nature makes it infeasible to deploy in a real-world scenario. Using it to warm-start the ACOPF improves its convergence speed by 3.8x and guarantees a solution. At a mean runtime of less than $4\,\mathrm{min}$ compared to ACOPF's $15\,\mathrm{min}$, it approaches feasibility to deploy it in the five-minute time horizon.

4 Conclusion

We have shown that a GNN model can be trained offline on power grid optimization results. While it finds solutions for four orders of magnitude faster than ACOPF, it can also warm-start an optimizer to guarantee correctness of the solution. This allows us to dispense with the currently-used linear approximations and accelerates ACOPF computation by a factor of 3.75. Contrary to a vanilla ACOPF, it is feasible to deploy.

However, these results should not be understood as more than a first proof-of-concept: (a) Deployed optimizers are more complex and problem-specific. (b) We rely on synthetic data, since actual grid data is not publically accessible. (c) Deploying these solutions into existing power grid infrastructure is a gigantic task both from an engineering and organizational perspective.

Nonetheless, wide-spread adoption of such a system could save billions of dollars per year (Cain et al. 2012), and reduce emissions in the order of a hundred $\mathrm{MtCO_2\text{-}eq./year}$.

References

Battaglia, Peter W. et al. (2018). “Relational inductive biases, deep learning, and graph networks”. In: *arXiv:1806.01261 [cs, stat]*. arXiv: `1806.01261`.

Bienstock, Daniel and Abhinav Verma (2015). “Strong NP-hardness of AC power flows feasibility”. In: *arXiv:1512.07315 [math]*. arXiv: `1512.07315`.

Birchfield, Adam B., Ti Xu, Komal Shetye, and Thomas Overbye (2018). “Building Synthetic Power Transmission Networks of Many Voltage Levels, Spanning Multiple Areas”. In: *Proceedings of the 51st Hawaii International Conference on System Sciences*.

Cain, Mary B., Richard P. ONeill, and Anya Castillo (2012). “History of Optimal Power Flow and Formulations”. In: *Federal Energy Regulatory Commission 1 (2012)*.

Coffrin, Carleton, Russell Bent, Kaarthik Sundar, Yeesian Ng, and Miles Lubin (2018). “PowerModels.jl: An Open-Source Framework for Exploring Power Flow Formulations”. In: *2018 Power Systems Computation Conference (PSCC)*.

Fey, Matthias and Jan E. Lenssen (2019). “Fast Graph Representation Learning with PyTorch Geometric”. In: *ICLR Workshop on Representation Learning on Graphs and Manifolds*.

Frank, Stephen and Steffen Rebennack (2016). “An introduction to optimal power flow: Theory, formulation, and examples”. In: *IIE Transactions 48.12 (2016)*.

Guha, Neel, Zhecheng Wang, and Arun Majumdar (2019). “Machine Learning for AC Optimal Power Flow”. In: *arXiv:1910.08842 [cs.LG]*. arXiv: `1910.08842`.

IPCC (2014). *Climate Change 2014: Synthesis Report. Contribution of Working Groups I, II and III to the Fifth Assessment Report of the Intergovernmental Panel on Climate Change*.

Paszke, Adam, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer (2017). “Automatic Differentiation in PyTorch”. In: *NeurIPS Autodiff Workshop*.

Wächter, Andreas and Lorenz T Biegler (2006). “On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming”. In: *Mathematical programming* 106.1, pp. 25–57.

Zaheer, Manzil, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Ruslan R. Salakhutdinov, and Alexander J. Smola (2017). “Deep Sets”. In: *Advances in Neural Information Processing Systems*, pp. 3391–3401.

Appendix

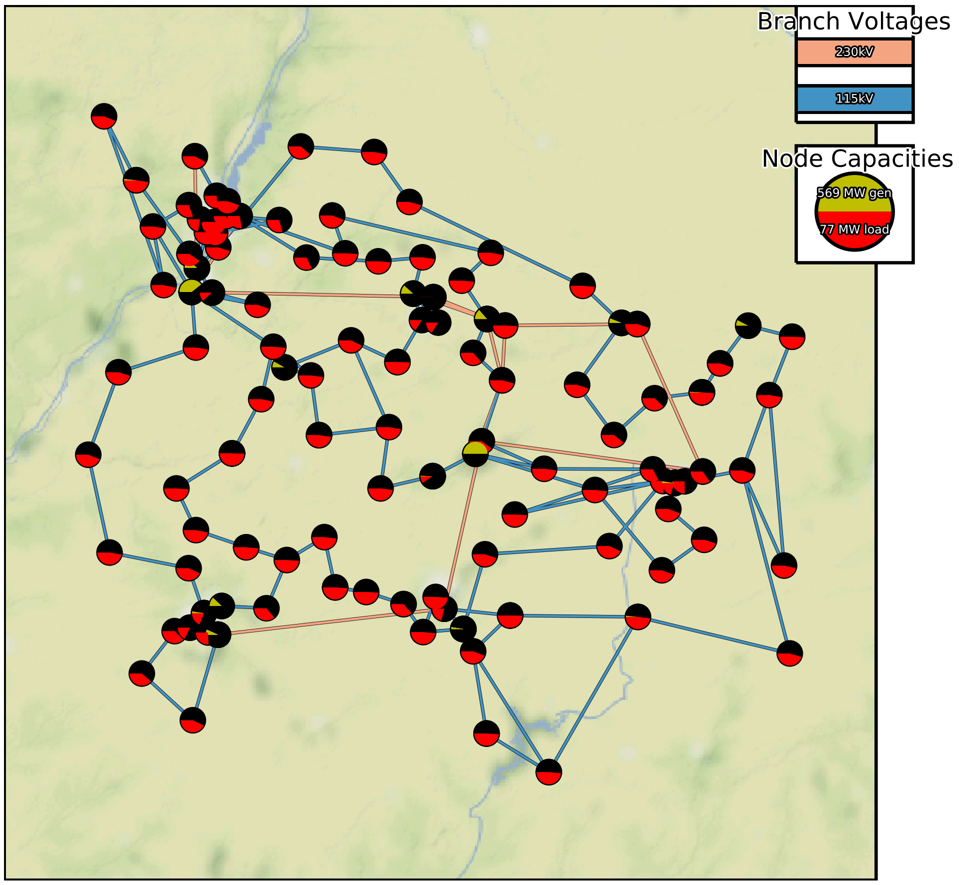

Figure 2: Map of the synthetic ILLINOIS dataset. Each bus is annotated with its connected maximum generation capacity (yellow) and load (red). Each branch is colour-coded according to its voltage.

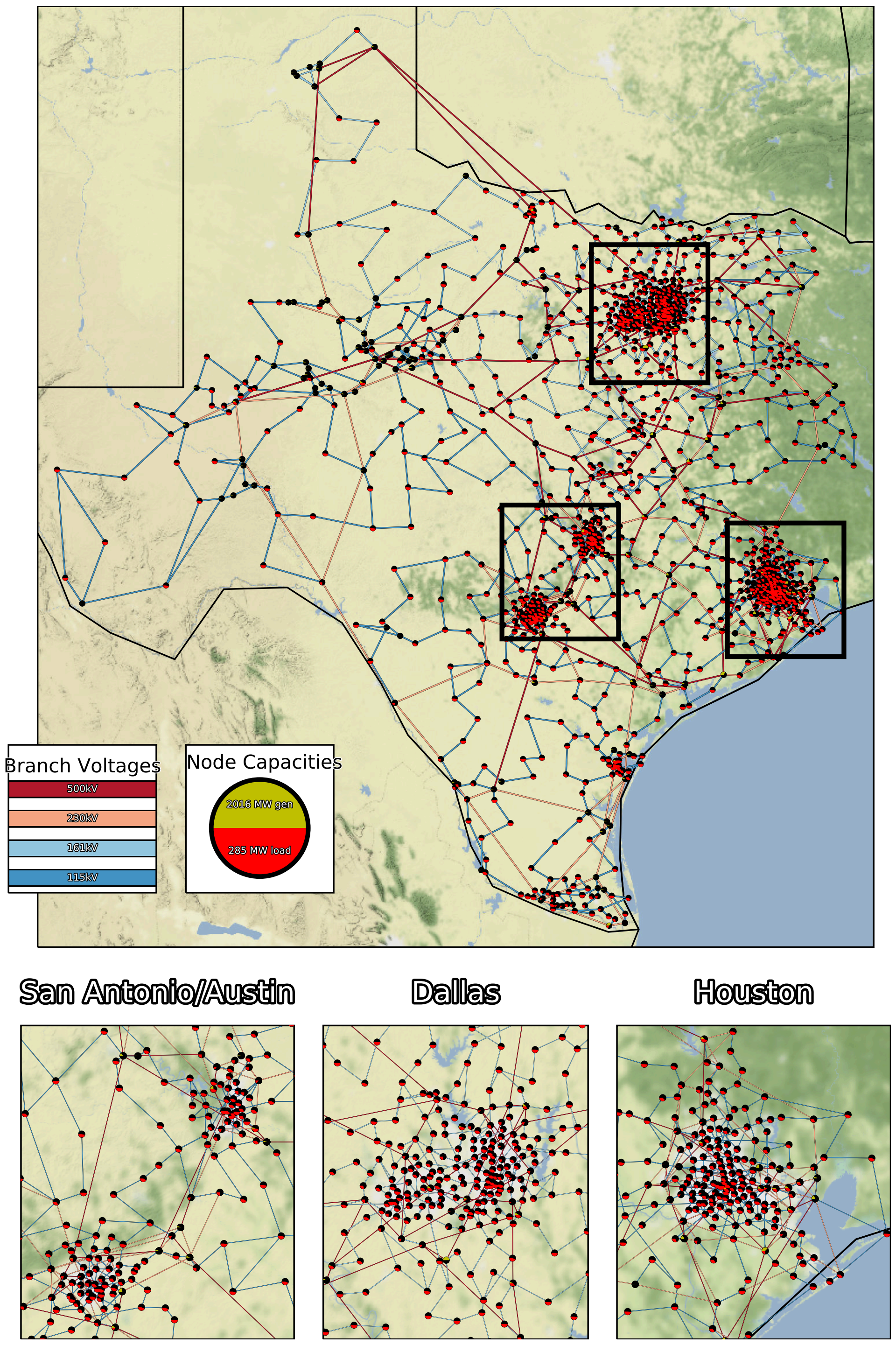

Figure 3: Map of the synthetic TEXAS dataset. For a description of the format, see Fig. 2