Transmission Constraint Screening for Production Cost Modeling at Scale

Bernard Knueven

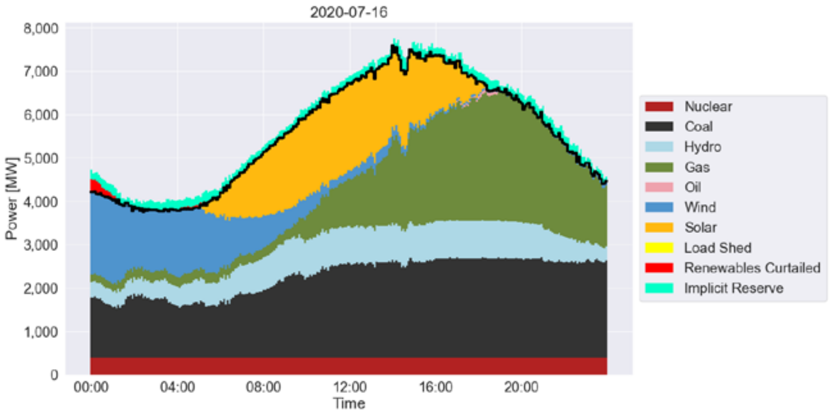
2022 INFORMS Annual Meeting

16 October 2022

Contents

Production Cost Modeling

- Simulate the large-scale electricity system (generation, load, transmission line flows) in 5-minute to 1-hour intervals for days, weeks, or a year
- Typically involves some sequence of Unit Commitment and Economic Dispatch problems



- Outputs: dispatch, transmission line flows, locational marginal prices (LMPs)
- Software:
  - Open Source: PowerSimulations.jl, Prescient
  - Propriety: PLEXOS, GridView, PROMOD
- Requires a robust unit commitment and economic dispatch engine

EGRET Overview

- EGRET – Electrical Grid Research and Engineering Tools
- Python-base package for electrical grid optimization built on the Pyomo algebraic modeling library
- Major Features:
  - Expression and solution of unit commitment problems
  - Expression and solution of economic dispatch and optimal power flow problems (e.g., DCOPF, ACOPF)
  - Library of formulations, approximations, and relaxations
  - Generic handling of data across model formulations and types
- EGRET serves as the unit commitment and economic dispatch engine for the Prescient Production Cost Modeling Engine
- EGRET is available under a BSD license at https://github.com/grid-parity-exchange/Egret

Transmission Constraints

- Transmission Constraints serve to limit the flow of electricity through a transmission line or transformer
  - More flow -> more heat -> line expansion and sagging
  - Transformers have their own power ratings for reliable operation
- In typical unit commitment and economic dispatch problems the line flow calculation used is a *linear approximation* of the AC power flow equations:

$$\sum_{l \in F^+(i)} f_l - \sum_{l \in F^-(i)} f_l = n_i \quad \forall i \in B$$

$$n_i = \sum_{g \in G(i)} p_g - L_i \quad \forall i \in B$$

$$f_l = B_l\left(\theta_{l(i)} - \theta_{l(j)}\right) \quad \forall l \in L$$

$$-F_l \leq f_l \leq F_l \quad \forall l \in L$$

$$\theta_{ref} = 0$$

Variables: $2|B| + |L| + |G|$

Equalities: $2|B| + |L| + 1$

Decision variables: p_g

Transmission Constraints

$$\sum_{l\in F^+(i)} f_l - \sum_{l\in F^-(i)} f_l = n_i \quad \forall i \in B$$

$$f_l = B_l\left(\theta_{l(i)} - \theta_{l(j)}\right) \quad \forall l \in L$$

$$-F_l \le f_l \le F_l \quad \forall l \in L$$

$$\theta_{ref} = 0$$

Rewrite in matrix notation:

- $\boldsymbol{A}$ is the $|L| \times (|B| - 1)$ incidence matrix
  - $a_{l,i} = 1$ if line l starts at bus i
  - $a_{l,i} = -1$ if line l ends at bus i
  - Remove the column corresponding to $\theta_{ref} = 0$
- $\boldsymbol{B_d}$ is a $|L| \times |L|$ diagonal matrix with B_l on the diagonals
- $\boldsymbol{\theta_B}$ is the vector of θ_i variables, $i \neq ref$
- $\boldsymbol{f_L}$ is the vector of f_l variables
- $\boldsymbol{n_B}$ is the vector of n_i variables, $i \neq ref$

$$\boldsymbol{n_B} = \boldsymbol{A}^T \cdot \boldsymbol{f_L}$$

$$\boldsymbol{1}^T\boldsymbol{n_B} + n_{ref} = 0$$

$$\boldsymbol{f_L} = \boldsymbol{B_d} \cdot \boldsymbol{A} \cdot \boldsymbol{\theta_B}$$

$$-\boldsymbol{F_L} \le \boldsymbol{f_L} \le \boldsymbol{F_L}$$

$$\boldsymbol{n_B} = \boldsymbol{A}^T \cdot \boldsymbol{B_d} \cdot \boldsymbol{A} \cdot \boldsymbol{\theta_B}$$

$$\boldsymbol{1}^T\boldsymbol{n_B} + n_{ref} = 0$$

$$\boldsymbol{f_L} = \boldsymbol{B_d} \cdot \boldsymbol{A} \cdot \boldsymbol{\theta_B}$$

$$-\boldsymbol{F_L} \le \boldsymbol{f_L} \le \boldsymbol{F_L}$$

Calculating Flows from $\boldsymbol{n_B}$

$$\boldsymbol{n_B} = \boldsymbol{A^T} \cdot \boldsymbol{B_d} \cdot \boldsymbol{A} \cdot \boldsymbol{\theta_B}$$
$$\boldsymbol{1}^T \boldsymbol{n_B} + n_{ref} = 0$$
$$\boldsymbol{f_L} = \boldsymbol{B_d} \cdot \boldsymbol{A} \cdot \boldsymbol{\theta_B}$$
$$-\boldsymbol{F_L} \leq \boldsymbol{f_L} \leq \boldsymbol{F_L}$$

$$\boldsymbol{1}^T \boldsymbol{n_B} + n_{ref} = 0$$
$$-\boldsymbol{F_L} \leq \boldsymbol{PTDF}^{L \times B} \cdot \boldsymbol{n_B} \leq \boldsymbol{F_L}$$

With:

$$n_i = \sum_{g \in G(i)} p_g - L_i \quad \forall i \in B$$

Variables: $|B| + |G|$

Equalities: $|B| + 1$

Given $\boldsymbol{\hat{n}_B}$ with $\boldsymbol{1}^T \boldsymbol{n_B} + n_{ref} = 0$:

- Solve $\boldsymbol{\hat{n}_B} = (\boldsymbol{A^T B_d A}) \cdot \boldsymbol{\theta_B}$ -> $\boldsymbol{\hat{\theta}_B}$
- $\boldsymbol{\hat{f}_L}$ <- $(\boldsymbol{B_d A}) \cdot \boldsymbol{\hat{\theta}_B}$

How to put in algebraic model?

- $\boldsymbol{\theta_B} = \left(\boldsymbol{A^T B_d A}\right)^{-1} \cdot \boldsymbol{n_B}$
- $\boldsymbol{f_L} = \underbrace{(\boldsymbol{B_d A}) \cdot \left(\boldsymbol{A^T B_d A}\right)^{-1}}_{\boldsymbol{PTDF}^{L \times B}} \cdot \boldsymbol{n_B}$

See Van den Bergh et al. (2014) for details

Comparing Models

$$\boldsymbol{n_B} = (\boldsymbol{A}^T \boldsymbol{B_d} \boldsymbol{A}) \cdot \boldsymbol{\theta_B}$$
$$\mathbf{1}^T \boldsymbol{n_B} + n_{ref} = 0$$
$$-\boldsymbol{F_L} \leq (\boldsymbol{B_d} \boldsymbol{A}) \cdot \boldsymbol{\theta_B} \leq \boldsymbol{F_L}$$

- Sparse if $\boldsymbol{A}$ is sparse
- $2|B| - 1$ variables
- $|B|$ equalities
- $|L|$ range constraints

- No computation to implement
- Solver must solve $\boldsymbol{n_B} = (\boldsymbol{A}^T \boldsymbol{B_d} \boldsymbol{A}) \cdot \boldsymbol{\theta_B}$ to calculate and enforce **any** line's flow

$$\mathbf{1}^T \boldsymbol{n_B} + n_{ref} = 0$$
$$-\boldsymbol{F_L} \leq \boldsymbol{PTDF}^{\boldsymbol{L \times B}} \cdot \boldsymbol{n_B} \leq \boldsymbol{F_L}$$

- Dense even if $\boldsymbol{A}$ is sparse
- $|B|$ variables
- 1 equality
- $|L|$ range constraints

- Need to calculate $\boldsymbol{PTDF}^{\boldsymbol{L \times B}}$
- Only need calculate rows of $\boldsymbol{PTDF}^{\boldsymbol{L \times B}}$ for **active** lines
- Roald & Molzahn (2019) show only a small subset (~1%) of these need to be enforce for a given load profile

Simple Algorithm for PTDF-model

- Calculate $\boldsymbol{PTDF}^{L\times B}$
- Initialize $L^A = \emptyset;\, viol \leftarrow True$
- While $viol$:
  - $\hat{\boldsymbol{n}}_{\boldsymbol{B}} \leftarrow$ Solve PTDF-DCOPF with L^A
  - Check for violations by calculating:
    - $\hat{\boldsymbol{f}}_{\boldsymbol{L}} \leftarrow \boldsymbol{PTDF}^{L\times B} \cdot \hat{\boldsymbol{n}}_{\boldsymbol{B}}$
  - $viol \leftarrow any(\hat{\boldsymbol{f}}_{\boldsymbol{L}} > \boldsymbol{F}_{\boldsymbol{L}}, \hat{\boldsymbol{f}}_{\boldsymbol{L}} < -\boldsymbol{F}_{\boldsymbol{L}})$
  - Update L^A by adding at least one violated line

PTDF-DCOPF

$$\min \textstyle\sum_{g\in G} c^g(p^g)$$

$$\underline{P}_g \le p_g \le \overline{P}^g \quad \forall g \in G$$

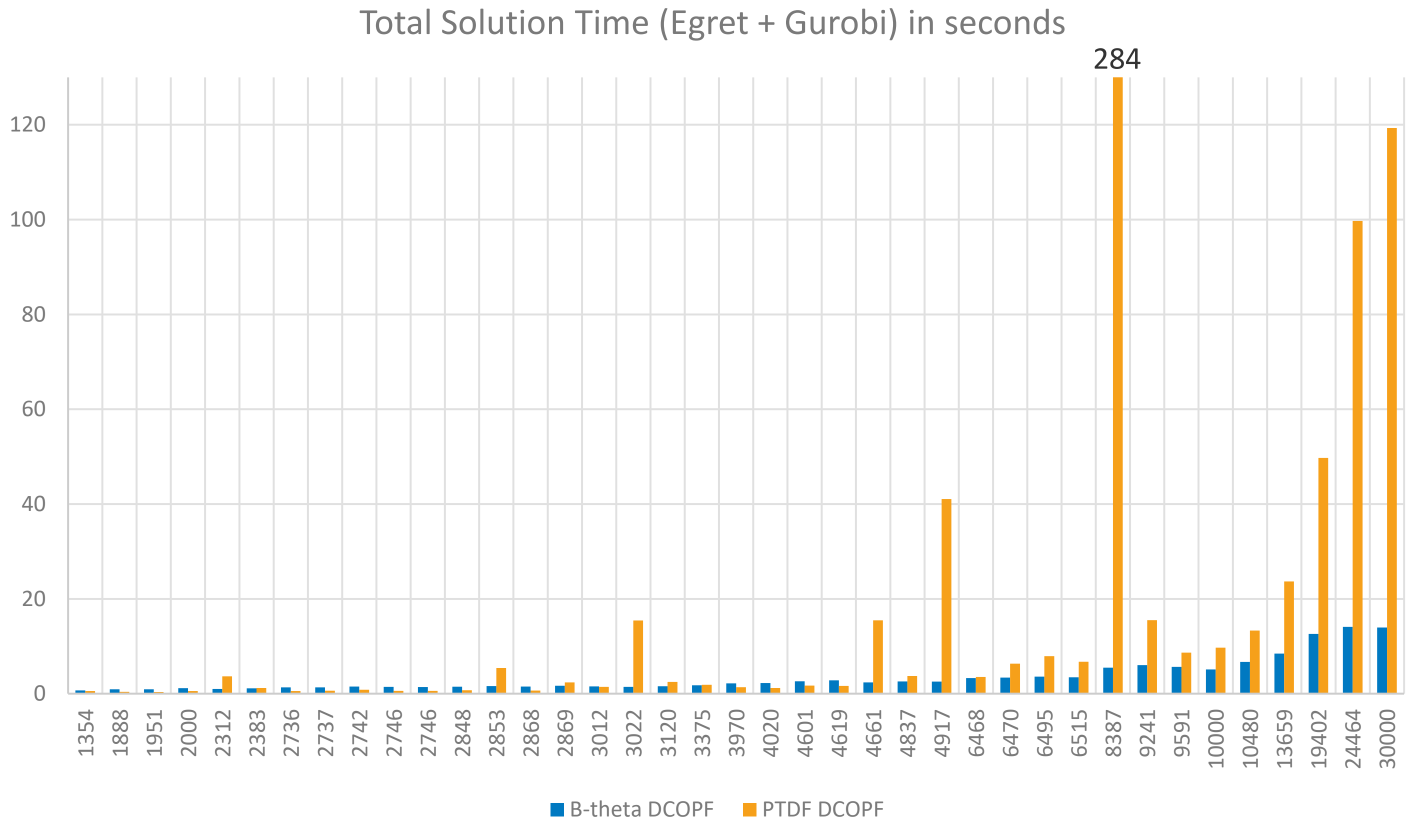
$$n_i = \sum_{g\in G(i)} p_g - L_i \quad \forall i \in B$$

$$\boldsymbol{1}^{\boldsymbol{T}}\boldsymbol{n}_{\boldsymbol{B}} + n_{ref} = 0$$

$$-F_l \le \boldsymbol{PTDF}^{l\times B} \cdot \boldsymbol{n}_{\boldsymbol{B}} \le F_l \quad \forall l \in L^A$$

This is basically the algorithm implemented by EGRET when the PTDF-DCOPF model was originally added in 2019

Simple Lazy-PTDF Model vs. B-theta



pglib-opf v21.07 (Babaeinejadsarookolaee, 2021) instances >1000 buses

Simple Algorithm for PTDF-model

- Calculate $\boldsymbol{PTDF}^{L\times B}$
- Initialize $L^A = \emptyset$; $viol \leftarrow True$
- While $viol$:
  - $\hat{\boldsymbol{n}}_{\boldsymbol{B}} \leftarrow$ Solve PTDF-DCOPF with L^A
  - Check for violations by calculating:
    - $\hat{\boldsymbol{f}}_{\boldsymbol{L}} \leftarrow \boldsymbol{PTDF}^{L\times B} \cdot \hat{\boldsymbol{n}}_{\boldsymbol{B}}$
  - $viol \leftarrow any(\hat{\boldsymbol{f}}_{\boldsymbol{L}} > \boldsymbol{F}_{\boldsymbol{L}}, \hat{\boldsymbol{f}}_{\boldsymbol{L}} < -\boldsymbol{F}_{\boldsymbol{L}})$
  - Update L^A by adding at least one violated line

PTDF-DCOPF

$$\min \sum_{g\in G} c^g(p^g)$$

$$\underline{P}_g \leq p_g \leq \overline{P}^g \quad \forall g \in G$$

$$n_i = \sum_{g\in G(i)} p_g - L_i \quad \forall i \in B$$

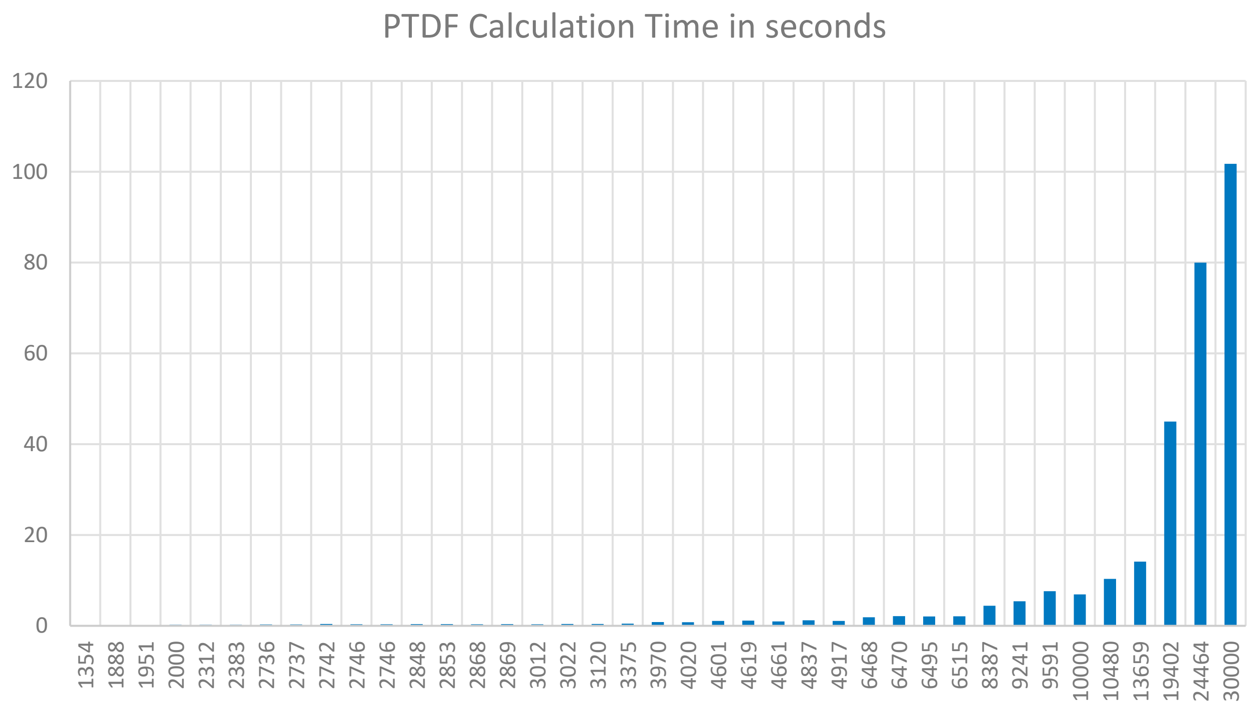
$$\boldsymbol{1}^T \boldsymbol{n}_{\boldsymbol{B}} + n_{ref} = 0$$

$$-F_l \leq \boldsymbol{PTDF}^{l\times B} \cdot \boldsymbol{n}_{\boldsymbol{B}} \leq F_l \quad \forall l \in L^A$$

$|B| \times |B|$ matrix inverse
$(|L| \times |B|)$ with $(|B| \times |B|)$ matrix-matrix multiplication

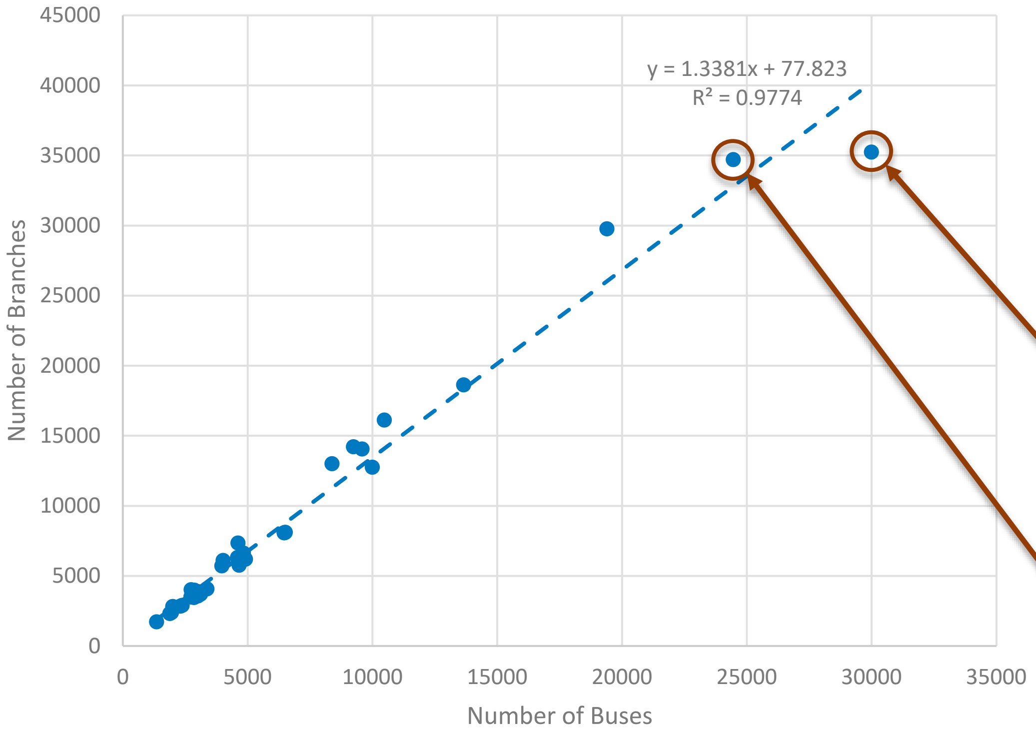
$(|L| \times |B|)$ with $|B|$ matrix-vector multiplication each loop

PTDF Calculation is Expensive



Becomes significant for “large-scale” systems (>10000 buses)

Storing $PTDF^{L \times B}$



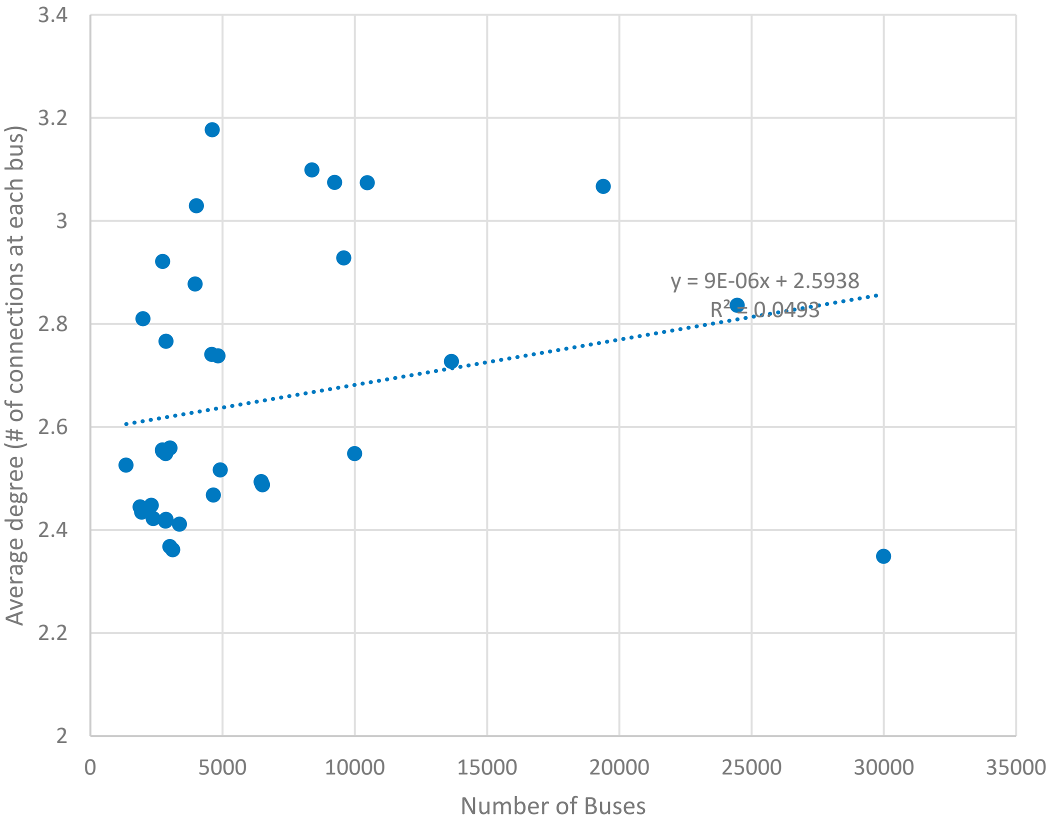
All networks in pglib-opf v21.07 (Babaeinejadsarookolaee, 2021):

- >1,000 buses (n=39)
- Number of branches increases linearly with number of buses

30000 buses X 35233 branches
-> ~7.9GB!
-> Invert a 30k X 30k matrix!

24464 buses X 34693 branches
-> ~6.3GB!
-> Invert a 24k X 24k matrix!

A is Sparse!



All networks in pglib-opf v21.07 (Babaeinejadsarookolaee, 2021):

- >1,000 buses (n=39), the average degree is 2.3 – 3.2
- The number of buses (nodes) does not seem to influence the average degree

This means $(A^T B_d A)$ is also (very) sparse – maintaining this sparsity is critical for performance

Revisiting the PTDF Calculation

$$\boldsymbol{n_B} = \boldsymbol{A}^T \cdot \boldsymbol{B_d} \cdot \boldsymbol{A} \cdot \boldsymbol{\theta_B}$$
$$\boldsymbol{1}^T \boldsymbol{n_B} + n_{ref} = 0$$
$$\boldsymbol{f_L} = \boldsymbol{B_d} \cdot \boldsymbol{A} \cdot \boldsymbol{\theta_B}$$
$$-\boldsymbol{F_L} \leq \boldsymbol{f_L} \leq \boldsymbol{F_L}$$

Given $\widehat{\boldsymbol{n}}_{\boldsymbol{B}}$ with $\boldsymbol{1}^T \boldsymbol{n_B} + n_{ref} = 0$:

- Solve $\widehat{\boldsymbol{n}}_{\boldsymbol{B}} = (\boldsymbol{A}^T \boldsymbol{B_d} \boldsymbol{A}) \cdot \boldsymbol{\theta_B}$ -> $\widehat{\boldsymbol{\theta}}_{\boldsymbol{B}}$
- $\widehat{\boldsymbol{f}}_{\boldsymbol{L}}$ <- $(\boldsymbol{B_d} \boldsymbol{A}) \cdot \widehat{\boldsymbol{\theta}}_{\boldsymbol{B}}$

So, we can calculate the flows without inverting $(\boldsymbol{A}^T \boldsymbol{B_d} \boldsymbol{A})$!

$$\boldsymbol{1}^T \boldsymbol{n_B} + n_{ref} = 0$$
$$-\boldsymbol{F_L} \leq \boldsymbol{PTDF}^{\boldsymbol{L \times B}} \cdot \boldsymbol{n_B} \leq \boldsymbol{F_L}$$

Recall $\boldsymbol{PTDF}^{\boldsymbol{L \times B}} := (\boldsymbol{B_d} \boldsymbol{A}) \cdot \left(\boldsymbol{A}^T \boldsymbol{B_d} \boldsymbol{A}\right)^{-1}$

Or $\boldsymbol{PTDF}^{\boldsymbol{L \times B}} \cdot \left(\boldsymbol{A}^T \boldsymbol{B_d} \boldsymbol{A}\right) = (\boldsymbol{B_d} \boldsymbol{A})$

So, given a line l, we can calculate a single PTDF row by solving $\left(\boldsymbol{A}^T \boldsymbol{B_d} \boldsymbol{A}\right)$ on the left!

$$\boldsymbol{PTDF}^{\boldsymbol{l \times B}} \cdot \left(\boldsymbol{A}^T \boldsymbol{B_d} \boldsymbol{A}\right) = (\boldsymbol{B_d} \boldsymbol{A})_l$$

Factorizing $A^T B_d A$

- The matrix $\boldsymbol{A^T B_d A}$ is sparse, symmetric, and (typically) positive semidefinite.
  - These facts suggest we should pre-compute a Cholesky (LL^T) or LDL^T factorization of $\boldsymbol{A^T B_d A}$
- However, while `scipy` (Virtanen, 2020) has Cholesky and LDL^T factorization routines, none are sparsity-preserving!
- `scipy` does, however, have a sparsity-preserving LU-factorization routine available, `SuperLU` (Li, 2005)
- `SuperLU` (and similar codes, e.g., `HSL_ma57`) have advanced pivoting methods to ensure sparsity in the original matrix is maintained in the factors.
- Instead of pre-computing $\boldsymbol{PTDF^{L\times B}}$, Egret instead computes a single, sparse, LU factorization of $\boldsymbol{A^T B_d A}$, utilizing `SuperLU`'s solve method for both computing $\boldsymbol{\theta_B}$ and $\boldsymbol{PTDF^{l\times B}}$ for need l

Sparsity-Preserving Algorithm for PTDF-model

- Factorize $\boldsymbol{A}^T\boldsymbol{B}_d\boldsymbol{A} = \mathbf{LU}$
- Initialize $L^A = \emptyset$; $viol \leftarrow True$
- While $viol$:
  - $\widehat{\boldsymbol{n}}_B \leftarrow$ Solve PTDF-DCOPF with L^A
  - Check for violations by calculating:
    - $\widehat{\boldsymbol{\theta}}_B \leftarrow \mathbf{LU}.\mathrm{solve}(\widehat{\boldsymbol{n}}_B)$
    - $\widehat{\boldsymbol{f}}_L \leftarrow (\boldsymbol{B}_d\boldsymbol{A}) \cdot \widehat{\boldsymbol{\theta}}_B$
  - $viol \leftarrow any(\widehat{\boldsymbol{f}}_L > \boldsymbol{F}_L, \widehat{\boldsymbol{f}}_L < -\boldsymbol{F}_L)$
  - Update L^A by adding at least one violated line
  - $\boldsymbol{PTDF}^{l \times B} \leftarrow \mathbf{LU}.\mathrm{solve}((\boldsymbol{B}_d\boldsymbol{A})_l, \text{'T'})$

PTDF-DCOPF

$$\min \sum_{g \in G} c^g(p^g)$$

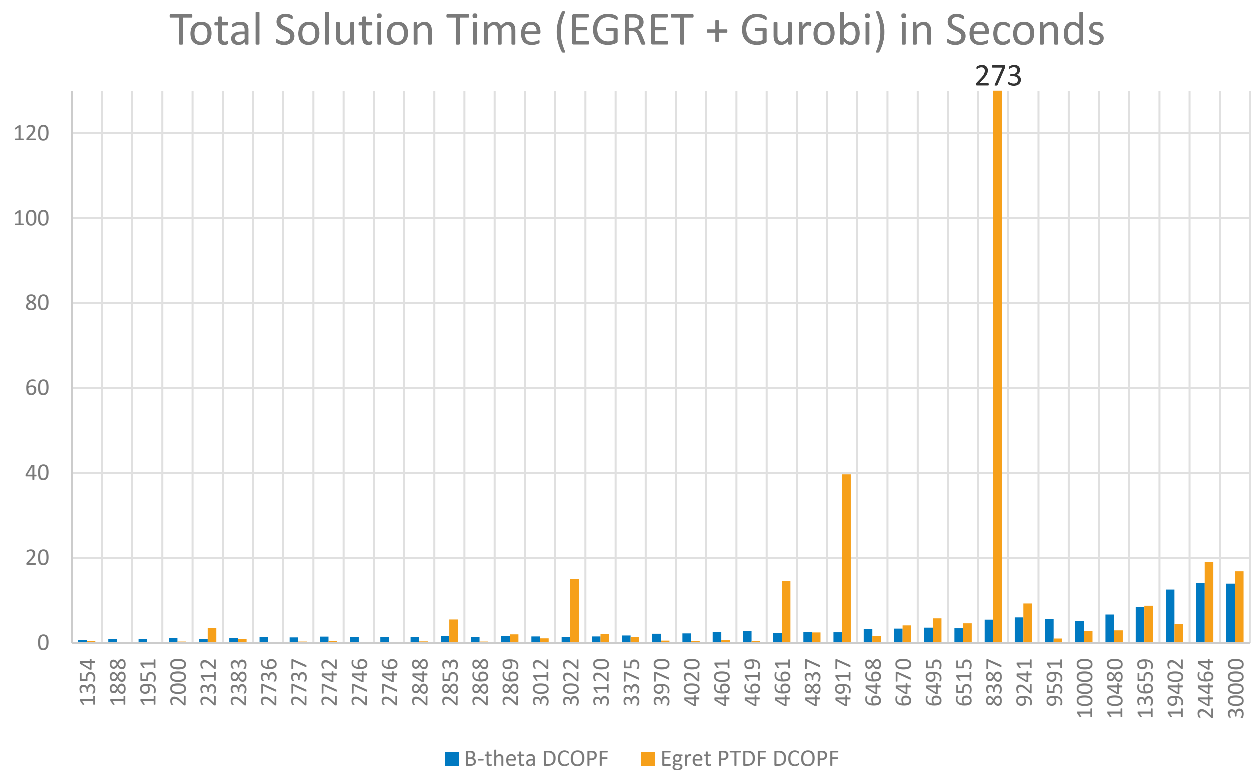
$$\underline{P}_g \le p_g \le \overline{P}^g \quad \forall g \in G$$

$$n_i = \sum_{g \in G(i)} p_g - L_i \quad \forall i \in B$$

$$\mathbf{1}^T \boldsymbol{n}_B + n_{ref} = 0$$

$$-F_l \le \boldsymbol{PTDF}^{l \times B} \cdot \boldsymbol{n}_B \le F_l \quad \forall l \in L^A$$

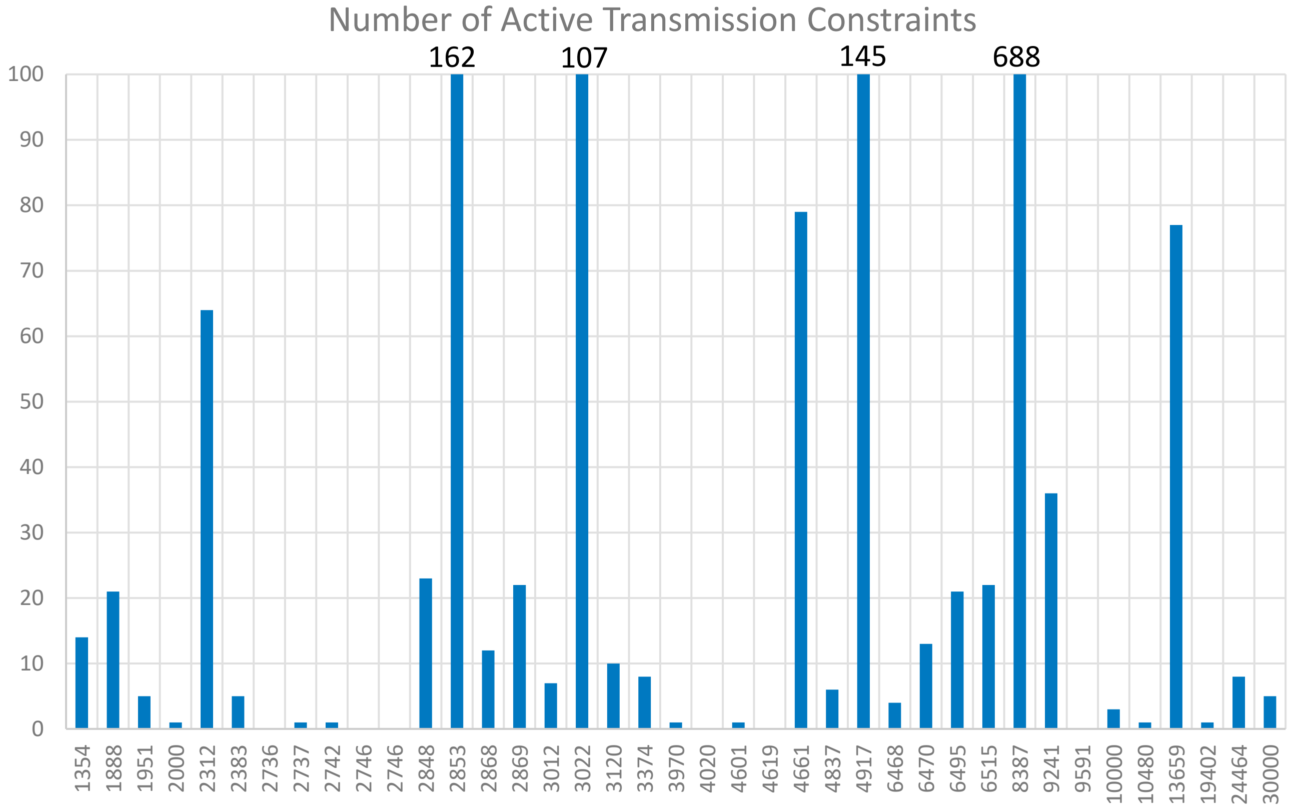
EGRET Lazy-PTDF Model vs. B-theta



Includes time in Pyomo

- Significant for building the B-theta DCOPF model & communicating it to Gurobi
- Significant for building individual PTDF-constraints for PTDF-DCOPF

Active Transmission Constraints



Performance outlier for EGRET's PTDF-DCOPF code, `pglib_opf_case8387_pegase.m`, has >5% of its transmission lines binding

Network Formulation has a big impact on PCM Runtimes

- Week-long simulation of the RTS-GMLC system using Prescient:
  - 73 buses
  - 120 branches
- XpressMP solver
- Solved Unit Commitment problems (7 total) to various MIP Gaps

MIP Gap	EGRET B-theta	EGRET Lazy PTDF	% Improvement
1.00%	252 s	213 s	15.4%
0.10%	311 s	231 s	25.8%
0.01%	552 s	257 s	53.5%
0.00%	621 s	336 s	45.9%

Conclusions

- The screening and calculation of transmission constraints is critical for Unit Commitment and helpful for DCOPF
  - Performance can be significantly improved with initial set of active constraints
- Maintain sparsity up until the point where a dense representation is required
- `scipy.sparse` has excellent tools for doing this, enabling a performant workflow in Python

References

Roald, Line A., and Daniel K. Molzahn. "Implied constraint satisfaction in power system optimization: The impacts of load variations." 2019 57th Annual Allerton Conference on Communication, Control, and Computing (Allerton). IEEE, 2019.

Van den Bergh, Kenneth, Erik Delarue, and William D'haeseleer. "DC power flow in unit commitment models." *KU Leuven TME WP EN2014-12.* (2014).

Babaeinejadsarookolaee, Sogol, et al. "The power grid library for benchmarking ac optimal power flow algorithms." *arXiv preprint arXiv:1908.02788* (2019).

Virtanen, Pauli, et al. "SciPy 1.0: fundamental algorithms for scientific computing in Python." *Nature methods* 17.3 (2020): 261-272.

Li, Xiaoye S. "An overview of SuperLU: Algorithms, implementation, and user interface." *ACM Transactions on Mathematical Software (TOMS)* 31.3 (2005): 302-325.

Q&A

www.nrel.gov

NREL/PR-2C00-84287

This work was authored by the National Renewable Energy Laboratory, operated by Alliance for Sustainable Energy, LLC, for the U.S. Department of Energy (DOE) under Contract No. DE-AC36-08GO28308. This work was conducted as part of the Design Integration and Synthesis Platform to Advance Tightly Coupled Hybrid Energy Systems (DISPATCHES) project through the Grid Modernization Lab Consortium with funding from the U.S. Department of Energy's Office of Fossil Energy and Carbon Management, Office of Nuclear Energy, and Hydrogen and Fuel Cell Technology Office. The views expressed in the article do not necessarily represent the views of the DOE or the U.S. Government. The U.S. Government retains and the publisher, by accepting the article for publication, acknowledges that the U.S. Government retains a nonexclusive, paid-up, irrevocable, worldwide license to publish or reproduce the published form of this work, or allow others to do so, for U.S. Government purposes. A portion of this research was performed using computational resources sponsored by the U.S. Department of Energy's Office of Energy Efficiency and Renewable Energy and located at the National Renewable Energy Laboratory.

NREL Transforming ENERGY