# Making Analysis of Power Systems Scalable through Distributed Computing

Aryan Ritwajeet Jha

# My interest in Distributed Computing

- Have always worked in Power System Control and Stability domain
- Transmission Systems are well-behaved, homogeneous

Illustrations by Pixeltrue on icons8

# The Average Transmission Line.

Illustrations by Pixeltrue on
icons8

# My interest in Distributed Computing

- Have always worked in Power System Control and Stability domain
- Transmission Systems are well-behaved, homogeneous
- Low R/X ratio.
- Good Jacobians
- Easy computations due to low variance in system components.

Illustrations by Pixeltrue on icons8

# My interest in Distributed Computing

- Now have been assigned to work on Distribution Systems.
- These lines are NOT homogeneous!

Illustrations by Pixeltrue on icons8

# The Average Distribution System at a feeder level substation in the subcontinent.

# My interest in Distributed Computing

- Now have been assigned to work on Distribution Systems.
- Highly heterogeneous!
- High R/X ratio
- NOT well-behaved Jacobians, YBus, etc.
- Unbalanced phases
- Our usual analysis methods will NOT work for these systems.
- Distributed Computing might help.

Illustrations by Pixeltrue on icons8

# What do I NOT mean by Distributed Computing?

# Fast Newton-Raphson Power Flow Analysis Based on Sparse Techniques and Parallel Processing

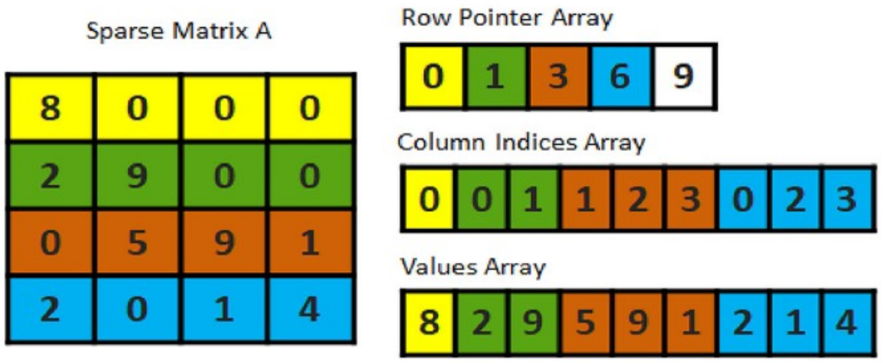
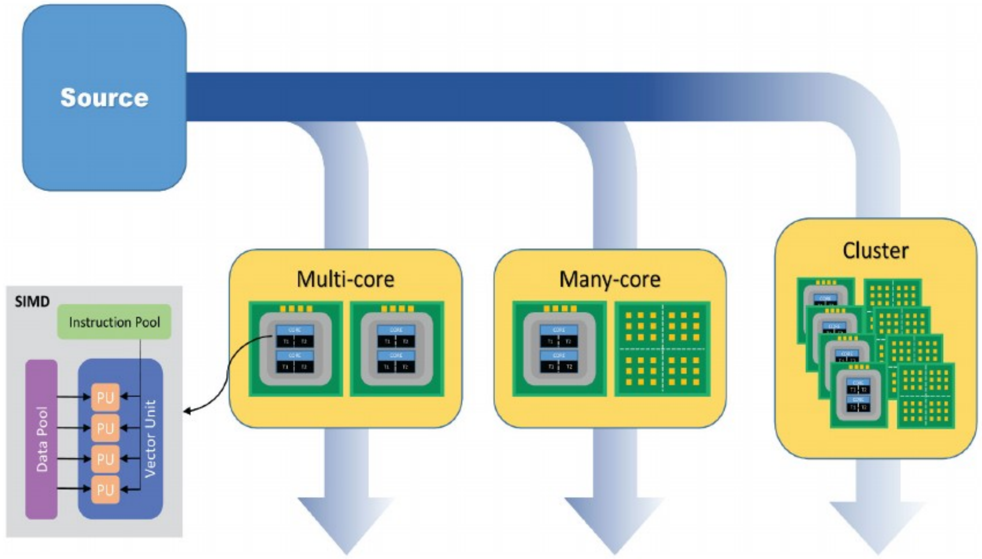



Fig. 1. A Sparse Matrix and its Corresponding CSR Arrays.

Sparsification reduces the data computation overhead and makes our analysis more scalable.

# Fast Newton-Raphson Power Flow Analysis Based on Sparse Techniques and Parallel Processing




Parallel Processing of Data using mutli-core CPUs, GPUs and SMIDs is also referred to as distributed computing.

# Fast Newton-Raphson Power Flow Analysis Based on Sparse Techniques and Parallel Processing

TABLE VII
PERFORMANCE COMPARISON OF IMPLEMENTED NEWTON-RAPHSON POWER FLOW WITH OTHER REFERENCES

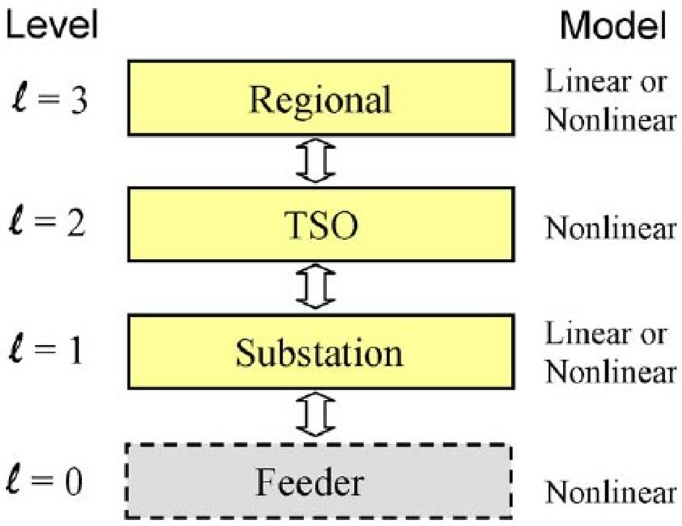
| Number of Buses | Reference | Platform | Runtime (Seconds) |
|---|---|---|---|
| 82,000 | Proposed | CPU-Sparse | **1.87** |
| | MATPOWER-NR | CPU-Sparse | 3.91 |
| | MATPOWER-NR-Qlim | CPU-Sparse | Diverged |
| | MATPOWER-FD | CPU-Sparse | 12.1 |

A speedier execution of data processing is certainly a desired output for a scalable algorithm.

But none of these methods refer to the usage of Distributed Computing here.

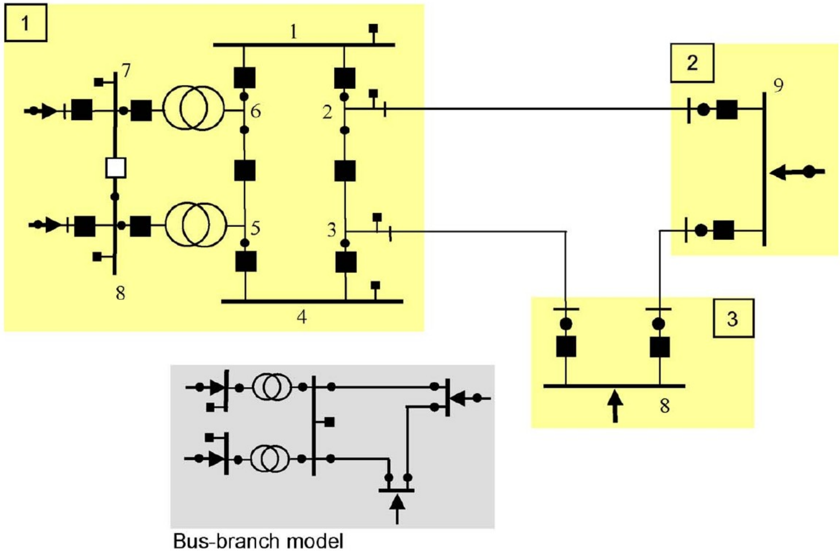
# Alright.. then what do I mean by Distributed Computing?

# A Multilevel State Estimation Paradigm for Smart Grids



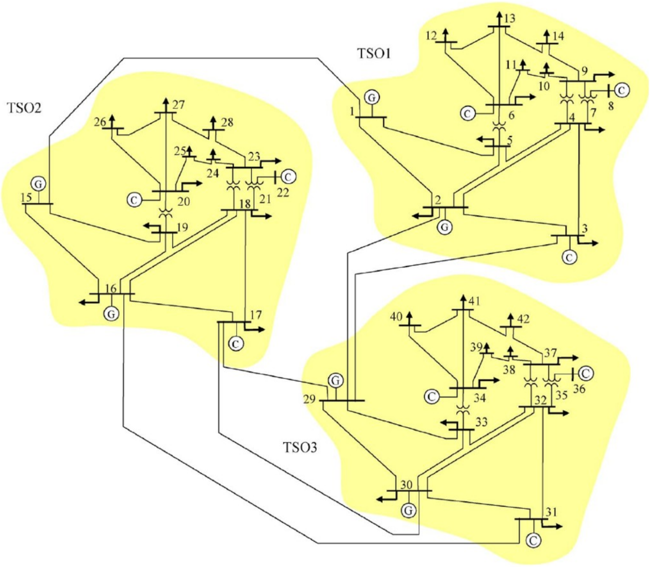

Heirarchial data computation and data reporting

# A Multilevel State Estimation Paradigm for Smart Grids




Dividing a central system into several subsystems and analyzing them all in parallel.

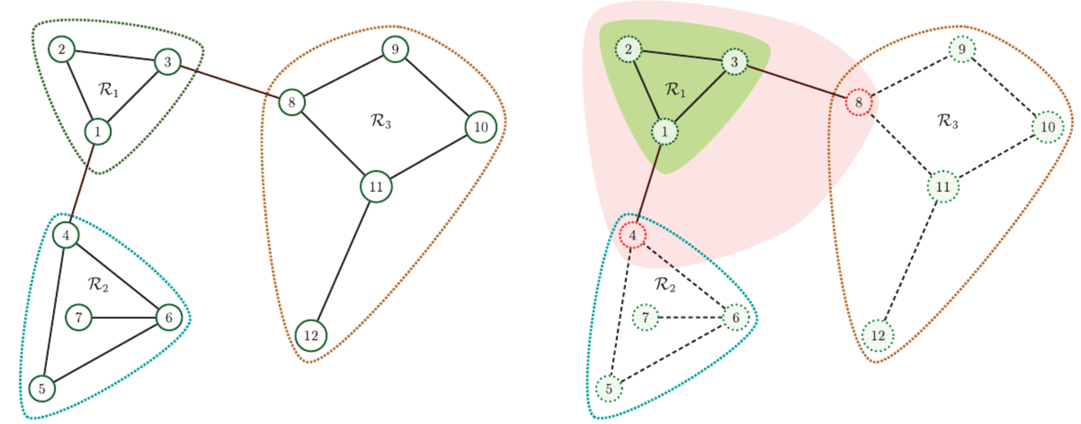
# A Multilevel State Estimation Paradigm for Smart Grids




Dividing a central system into several subsystems and analyzing them all in parallel. Here: Transmission Systems.

# Use Distributed Computing because you should.

# Distributed power flow and distributed optimization—Formulation, solution, and open source implementation




a) Example how to decompose a power grid into three regions $\{1, 2, 3\}$, $\{4, 5, 6, 7\}$, and $\{8, 9, 10, 11, 12\}$.

b) From the perspective of region $\mathcal{R}_1$, the core buses are buses $\{1, 2, 3\}$, and the copy buses are buses $\{4, 8\}$.

**Fig. 1.** Graphical depiction of nomenclature for distributed power flow problems, see Section 2.1.

Data privacy issues due to oversharing between different TSOs. How to counter that?

# Distributed power flow and distributed optimization—Formulation, solution, and open source implementation

### 2.3. *Distributed optimization problem*

The distributed power flow problem from (4) is a system of nonlinear equations—in a form amenable to distributed optimization. We propose to solve Problem (4) either as a *distributed feasibility problem*

$$\min_{\substack{x_i, z_i \\ \forall i \in \{1,\dots,n^{\text{reg}}\}}} \quad 0 \quad \text{s.t.} \tag{6a}$$

$$g_i^{\text{pf}}(x_i, z_i) = 0 \tag{6b}$$

$$g_i^{\text{bus}}(x_i) = 0 \tag{6c}$$

$$\sum_{i=1}^{n^{\text{reg}}} A_i \begin{bmatrix} x_i \\ z_i \end{bmatrix} = 0, \tag{6d}$$

or as a *distributed least-squares problem*[10]

$$\min_{\substack{x_i, z_i \\ \forall i \in \{1,\dots,n^{\text{reg}}\}}} \sum_{i=1}^{n^{\text{reg}}} \left\| \begin{bmatrix} g_i^{\text{pf}}(x_i, z_i) \\ g_i^{\text{bus}}(x_i) \end{bmatrix} \right\|^2 \quad \text{s.t.} \quad \sum_{i=1}^{n^{\text{reg}}} A_i \begin{bmatrix} x_i \\ z_i \end{bmatrix} = 0. \tag{7}$$

Solve for OPF separately for every single sub-system.

**Algorithm 1** ADMM for problem (8)

**Initialization:** $\zeta_i^0, \lambda_i^0$ **for all** $i \in \mathcal{R}$, $\rho$
**Repeat:**

1) $\chi_i^{k+1} = \underset{g_i(\chi_i)=0}{\operatorname{argmin}} f_i(\chi_i) + \lambda_i^{k\top} A_i \chi_i + \frac{\rho}{2} \|A_i(\chi_i - \zeta_i^k)\|_2^2, \quad i \in \mathcal{R}$ (parallel)
2) $\zeta^{k+1} = \underset{A\zeta=0}{\operatorname{argmin}} \sum_{i \in \mathcal{R}} -\lambda_i^{k\top} A_i \zeta_i + \frac{\rho}{2} \|A_i(\chi_i^{k+1} - \zeta_i)\|_2^2$ (centralized)
3) $\lambda_i^{k+1} = \lambda_i^k + \rho A_i(\chi_i^{k+1} - \zeta_i^{k+1}), \quad i \in \mathcal{R}$ (parallel)

**Algorithm 2** ALADIN for problem (8)

**Initialization:** $\zeta_i^0, \lambda^0, \Sigma_i \succ 0$ **for all** $i \in \mathcal{R}$, $\nu, \rho$
**Repeat:**

1) Solve for all $i \in \mathcal{R}$

$$\chi_i^k = \underset{g_i(\chi_i)=0}{\operatorname{argmin}} f_i(\chi_i) + \lambda^{k\top} A_i \chi_i + \frac{\nu}{2} \|\chi_i - \zeta_i^k\|_{\Sigma_i}^2, \quad \text{(parallel)}$$

2) Compute $\nabla f_i(\chi_i^k)$, $B_i^k \approx \nabla_{\chi_i}^2 \left(f_i(\chi_i^k) + \gamma_i^\top g_i(\chi_i^k)\right)$, $\nabla g_i(\chi_i^k)$.
3) Solve the coordination QP

$$\Delta\chi^k = \underset{\Delta\chi}{\operatorname{argmin}} \sum_{i \in \mathcal{R}} \tfrac{1}{2} \Delta\chi_i^\top B_i^k \Delta\chi_i + \nabla f_i^\top(\chi_i^k) \Delta\chi_i \quad \text{(centralized)}$$
$$+ \lambda^{k\top} \left(\textstyle\sum_{i \in \mathcal{R}} A_i(\chi_i^k + \Delta\chi_i) - b\right) + \frac{\rho}{2} \|\textstyle\sum_{i \in \mathcal{R}} A_i(\chi_i^k + \Delta\chi_i) - b\|_2^2$$
$$\text{subject to } \nabla g(\chi^k)\Delta\chi = 0.$$

4) Set $\zeta_i^{k+1} = \chi_i^k + \Delta\chi_i^k$ and $\lambda^{k+1} = \lambda^k + \rho\left(\sum_{i \in \mathcal{R}} A_i \chi_i - b\right)$. (parallel)

# Distributed power flow and distributed optimization—Formulation, solution, and open source implementation

**Table 4**
Computing times for different test cases and different solvers when solving the distributed least-squares problem (7) with ALADIN and sensitivities from rapidPF .

| Buses | $n^{reg}$ | MATPOWER case files | $n^{conn}$ | Solution time in s for | | | ALADIN iterations |
|---|---|---|---|---|---|---|---|
| | | | | fminunc | fmincon | worhp | |
| 53 | 3 | 9, 14, 30 | 3 | 2.5 | 2.2 | 2.4 | 4 |
| 354 | 3 | 3 × 118 | 5 | 2.5 | 3.1 | 4.8 | 5 |
| 418 | 2 | 118, 300 | 2 | 4.5 | 5.2 | 7.0 | 5 |
| 826 | 7 | 7 × 118 | 7 | 3.7 | 5.3 | 7.2 | 5 |
| 1180 | 10 | 10 × 118 | 11 | 4.9 | 6.7 | 9.8 | 6 |
| 2708 | 2 | 2 × 1354 | 1 | 212.7 | 41.9 | 53.6 | 4 |
| 4662 | 5 | 3 × 1354, 2 × 300 | 4 | 387.9 | 90.1 | 113.8 | 5 |

The authors designed an in-house algorithm acronymized ALADIN which performed

# Use Distributed Computing because you MUST.

# Distributed Optimization Using Reduced Network Equivalents for Radial Power Distribution Systems

## Distributed Computing for Scalable Optimal Power Flow in Large Radial Electric Power Distribution Systems with Distributed Energy Resources

**Algorithm 1:** Distributed Algorithm for Scaled OPFs

1 Decompose the network into $N$ areas, so that each area has a maximum specified node numbers
2 Initialize complicating variables, $\mathbf{Y}^0 \in \mathcal{S}$; error, $e = 1$; and macro-iteration count $n = 0$
3 If $|e| \leq \epsilon_{tol}$, stop iteration count, and go to step 10
4 Else, increase iteration count $n$: $n \leftarrow n + 1$
5 Solve $\Phi_m$ in parallel using **Algorithm 2**, for all decomposed areas $A_m$, where, $\Phi_m$ depicts the sub-problem –
$$\Phi_m : \; X_m^{(n)} := \underset{X_m \in S_m}{\text{argmin/argmax}} \; f_m\left(X_m, y_{m'}^{(n-1)}\right)$$
6 Update all the complicating variables, $\mathbf{Y}$, using (9), where $\alpha$ can be constant or adaptive
7 Check residual vector $\mathcal{R}^{(n)} = \left[\; \mathbf{Y}^{(n)} - \mathbf{Y}^{(n-1)} \;\right]$
8 $e = \max\, |\mathcal{R}^{(n)}|$
9 Go to step 2
10 Return Global Minimizer:
$X^* = \{X_m^{(n)} \mid m = 1, 2, ..., N\}$

The authors wanted to test Distributed algorithms against Centralized algorithms for Optimal Power Flow for large distribution system with significant renewable penetration.

# Distributed Optimization Using Reduced Network Equivalents for Radial Power Distribution Systems

## Distributed Computing for Scalable Optimal Power Flow in Large Radial Electric Power Distribution Systems with Distributed Energy Resources

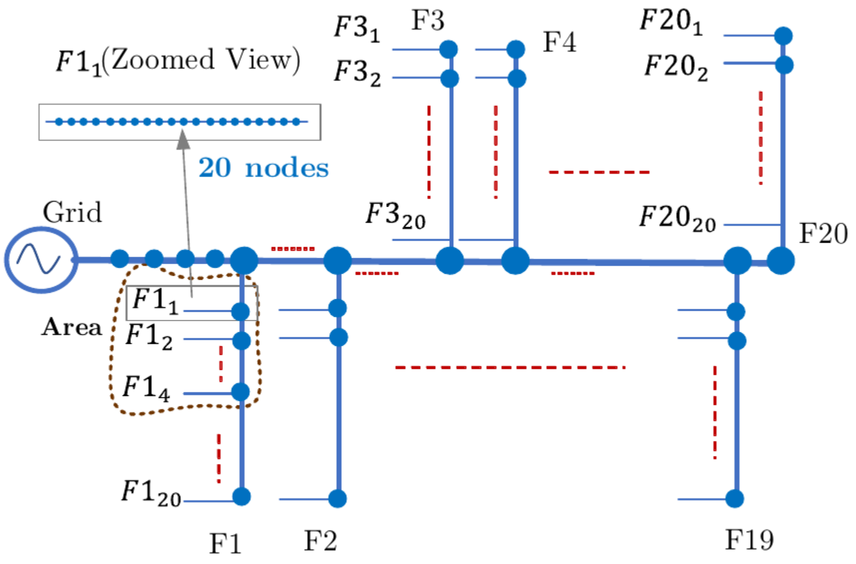
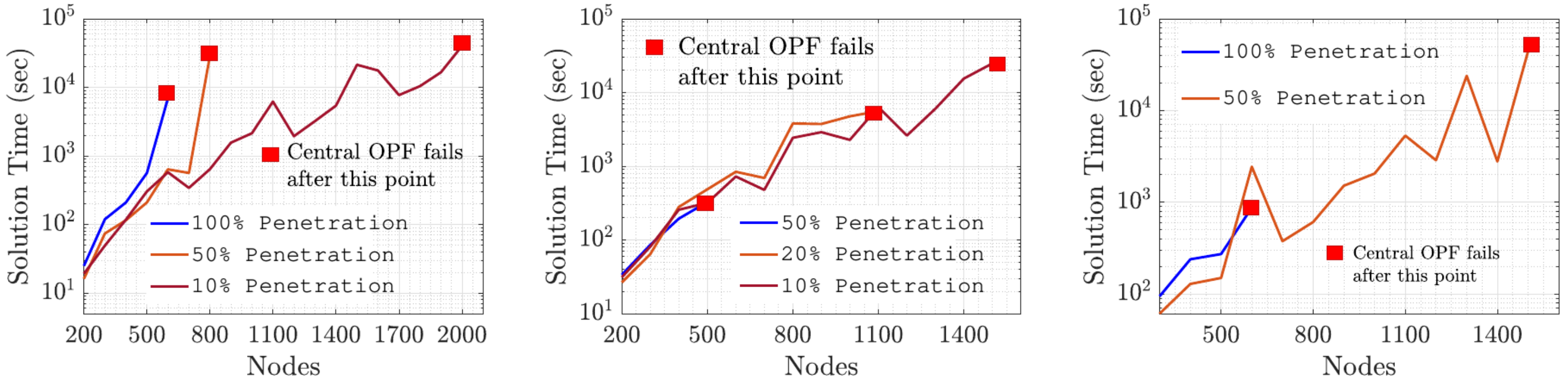



Figure 1: Synthetic 10,000 Node System

They used synthetic feeder-level distribution systems with significant DER penetration and applied their grid decomposition algorithm before applying their Distributed OPF algorithm.

# Distributed Optimization Using Reduced Network Equivalents for Radial Power Distribution Systems

## Distributed Computing for Scalable Optimal Power Flow in Large Radial Electric Power Distribution Systems with Distributed Energy Resources




(a) Loss minimization objective (b) DER maximization objective (c) ΔV minimization objective

Figure 5: Solution Time for Central OPF (C-OPF) Problems for Different Sizes of Networks

Centralized OPF algorithms are not only being slower, they are failing to converge!

# What did we do today? A super short summary of the contents of the papers.

01

**Paper 1: Clarification on the term Distributed Computing**

Parallel processing is a part of Distributed Computing, but NOT the concept itself!

02

**Papers 2 and 3: Need for Distributed Computing**

Privacy.
Computational attractiveness.

03

**Papers 4 and 5: A dire need for Distributed Computing**

Larger systems with significant DER penetrations would simply NOT converge for Centralized OPF algorithms. Must use Distributed OPF algorithms.

# Where does all of this fit with the course work?

# In our EE 521 Classes, we learnt..

**Course work covered algorithms designed for Centralized Computation.**

Very effective for homogeneous systems which are NOT large. Robust control.

Lagrangian and Dual Problem Formulation

LP
Simplex Method
Interior Point Method
NLP
Quadratic Programming

**All of these methods are JUST as valid in Distributed Computing .. provided some adjustments are made.**

# Thank you.

# Making Analysis of Power Systems Scalable through Distributed Computing

Aryan Ritwajeet Jha