Approximation algorithms for power network optimization

Samuel Talkington

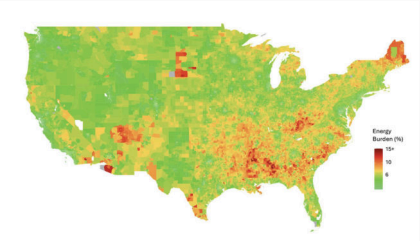
Harvard University

UVM



Harvard John A. Paulson
School of Engineering
and Applied Sciences

Electricity is becoming unaffordable as demand surges



Pricing growth has unequal impacts.[a]

- Average US electricity bills **rose 4.8% per year** from 2019–2023[a]
- **1 out of 3 US households:** forgoing basic necessities (food/medicine) to pay energy bills in 2024[b]
- US utility **consumer debt totaled $21.0 billion** as of September 2024.

[a] C. Batlle et al., "US federal resource allocations are inconsistent with concentrations of energy poverty," *Sci. Adv.* **10**, 2024.

[a] S. Forrester, et al., "Retail Electricity Price and Cost Trends," Lawrence Berkeley National Laboratory, 2024.

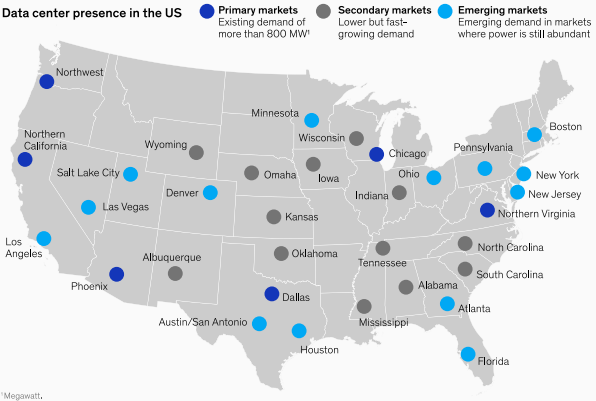
[b] U.S. Census Phase 4.2 Cycle 09 Household Pulse Survey: August 20 – September 16, 2024.

Graph algorithms for the power grid

Electricity demand is growing, and the grid must expand.

Operation and planning require graph structure.

Data centers are emerging in more remote locations, where power is still abundant and grids less strained.



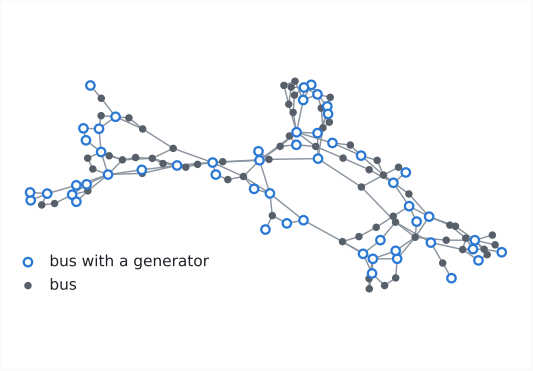
[1]Megawatt.
Source: Datacenters.com; S&P Global Market Intelligence 451 Research; McKinsey Data Center Demand model

McKinsey & Company

AC power flow equations

- A grid is an undirected graph $G = (\mathcal{N}, \mathcal{E})$: n nodes, m lines
- **Voltage phasors** $\boldsymbol{u} = \boldsymbol{v}\angle\boldsymbol{\theta} \in \mathbb{C}^n$;
- complex power **injections** $\boldsymbol{s} = \boldsymbol{p} + \mathrm{j}\boldsymbol{q} \in \mathbb{C}^n$
- **Admittance matrix** $\boldsymbol{Y} = \boldsymbol{G} + \mathrm{j}\boldsymbol{B} \in \mathbb{C}^{n\times n}$: a complex-valued graph Laplacian matrix[a]

[a] up to diagonal perturbations



$$\boldsymbol{s} = \operatorname{diag}(\boldsymbol{u})\,\underline{\boldsymbol{Y}\boldsymbol{u}}$$

power injections, topology; nonlinear: couples every node

Linearizing the power flow equations

Linear power flow model

A first-order Taylor expansion at *nominal* conditions ($\boldsymbol{u} = \mathbf{1}$) gives

$$\begin{bmatrix} p \\ q \end{bmatrix} \approx \begin{bmatrix} G & -B \\ -B & -G \end{bmatrix} \begin{bmatrix} v - 1 \\ \theta \end{bmatrix} \iff \begin{bmatrix} v - 1 \\ \theta \end{bmatrix} \approx \begin{bmatrix} R & X \\ X & -R \end{bmatrix} \begin{bmatrix} p \\ q \end{bmatrix}, \tag{1}$$

where $G \succ \mathbf{0}$, $B \prec \mathbf{0}$ are the real and imaginary components of the (reduced) admittance matrix Y, and $R, X \succ 0$ are the *resistance and reactance* matrices.

DC approximation: $G = \mathbf{0}$ and no shunts give $p \approx L\theta$, with:

$L := -B := AWA^T$ the negative susceptance (Laplacian) matrix,

A the incidence matrix (column a_e for line e) and $W = \mathrm{diag}(w)$ the line susceptances.

DC optimal power flow (DC-OPF)

$$\min_{g,\theta} \; c(g) \qquad \text{total operational cost}$$

$$\text{s.t.} \quad L\theta = p := Cg - d \qquad \text{DC power flow; } C\text{: generator-to-node incidence matrix}$$

$$\theta_r = 0 \text{ (or } \theta \perp \mathbf{1}) \qquad \text{“slack bus”/angle reference}$$

$$|w_e a_e^\top \theta| \le \overline{f}_e \qquad \text{limits on the flow } f_e = w_e a_e^\top \theta \text{ of every line } e$$

$$\underline{g} \le g \le \overline{g} \qquad \text{limits of every generator}$$

How fast can we solve this?

Stott, Jardim, Alsaç TPWRS'09.

Laplacian solvers in nearly linear time

Fast DC power flow

For $\boldsymbol{L}$ with $\mathrm{nnz}(\boldsymbol{L}) = \mathcal{O}(m)$, there is an algorithm Solve$(\boldsymbol{L}, \boldsymbol{p}, \epsilon)$ returns $\hat{\boldsymbol{\theta}}$ with

$$\left|\left|\hat{\boldsymbol{\theta}} - \boldsymbol{L}^{+}\boldsymbol{p}\right|\right|_{\boldsymbol{L}} \leq \epsilon \left|\left|\boldsymbol{L}^{+}\boldsymbol{p}\right|\right|_{\boldsymbol{L}}$$

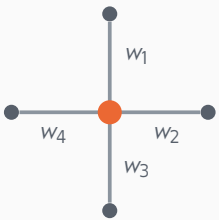
w.p. $\geq 1 - \mathcal{O}(\delta)$, in time $\mathcal{O}\left(m \log^2(1/\delta) \log n \log(1/\epsilon)\right) = \tilde{\mathcal{O}}(m)$.

Idea: Build a random Cholesky factor $\boldsymbol{C}$ such that, w.h.p. $\frac{1}{2}\boldsymbol{L} \preceq \boldsymbol{C}\boldsymbol{C}^{\star} \preceq \frac{3}{2}\boldsymbol{L}$.

Kyng, Sachdeva FOCS'16.

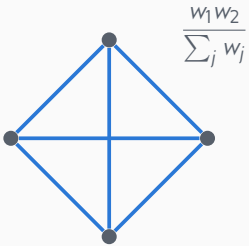
Elimination is the star-mesh transform

eliminate a node



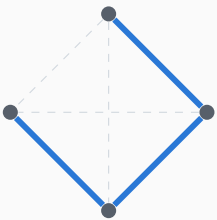
a Schur complement

a clique appears



Cholesky stores every clique

or a sample of it



approximate Cholesky: a few edges, unbiased

Rosen J. IEE'24; Kron '39; Kyng, Sachdeva FOCS'16.

Solving DC-OPF by constraint generation

factor $\boldsymbol{L}$ ▷ reference node removed

$\mathcal{M} \leftarrow \emptyset$ ▷ lines whose limit is enforced

repeat

- $\boldsymbol{g} \leftarrow$ DC-OPF with only the limits of the lines in $\mathcal{M}$
- $\boldsymbol{\theta} \leftarrow \boldsymbol{L}^{+}\boldsymbol{p}$ ▷ 1 solve
- $\mathcal{V} \leftarrow \{e : |w_e \boldsymbol{a}_e^{\mathsf{T}}\boldsymbol{\theta}| > \bar{f}_e\}$
- compute $\boldsymbol{L}^{+}\boldsymbol{a}_e$ for $e \in \mathcal{V}$ ▷ $|\mathcal{V}|$ solves
- $\mathcal{M} \leftarrow \mathcal{M} \cup \mathcal{V}$

until $\mathcal{V} = \emptyset$

Kelley J. SIAM'60; Peña-Ordieres, Molzahn, Roald, Wächter TPWRS'21.

- Flow on line e: $w_e \boldsymbol{a}_e^{\mathsf{T}}\boldsymbol{L}^{+}\boldsymbol{p}$, linear in $\boldsymbol{g}$, as $\boldsymbol{p} = \boldsymbol{C}\boldsymbol{g} - \boldsymbol{d}$.
- The subproblem is a QP with 1 constraint/line in $\mathcal{M}$.
- The added limits are cutting planes.

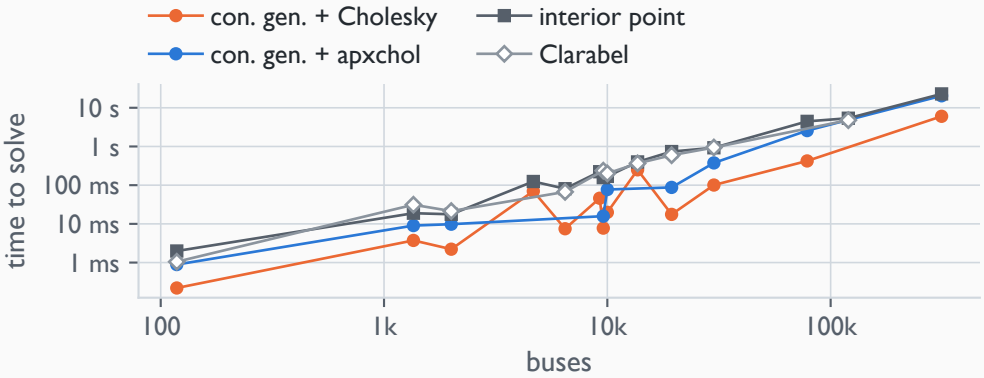
DC-OPF: after a few solves, Cholesky of L is cheaper than apxchol

Cost of S solves with one matrix: one factorization plus S solves.



Kyng, Sachdeva FOCS'16; Gao, Kyng, Spielman SISC'26; apxchol: Kulkov, Baumann, Kyng, software '26.

DC-OPF end to end: apxchol is slower than Cholesky



Benchmark networks and 2×2 copies of the two largest.

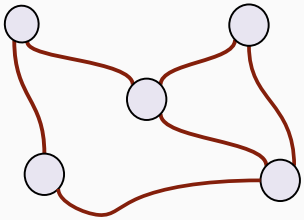
Benchmark networks: Babaeinejadsarookolaee et al. arXiv'19.

For DC-OPF, exact Cholesky is faster.

L never changes, and its Cholesky factor is sparse.

Planning solves with a new matrix every iteration.

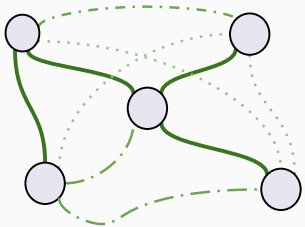
Loaded dice: The key insight



How should I connect the network?

What (single) config most reduces congestion?

Network Reconfiguration: Challenging, poorly scalable integer program



How should I connect the network?

What *family* of configs most reduces congestion?

Optimized switching probability

High Medium Low

Observation: *What if we didn't have to commit to just one configuration?*

Key idea: Optimize over a probability distribution of configurations → samples *concentrate* around "good" solutions → families of near-optimal designs.

Network reconfiguration problem

Network Reconfiguration

Find network configuration $\boldsymbol{s}$ that minimizes congestion:

$$\min_{\boldsymbol{s}\in\{0,1\}^m} \varphi(\boldsymbol{s}) \quad \text{s.t.} \quad \boldsymbol{s} \in \mathcal{S}$$

- Mixed-integer non-linear program
- **Hard in practice**, too

Example: $\varphi(\boldsymbol{s}) = \boldsymbol{p}^{\mathsf{T}} L_{\boldsymbol{s}}^{+} \boldsymbol{p}$ and

$$\begin{aligned}\mathcal{S} &= \{\boldsymbol{s} : \text{at most } q \text{ lines closed}\} \\ &= \{\boldsymbol{s} \in \{0,1\}^m \ : \ \|\boldsymbol{s}\|_1 \leq q\}\end{aligned}$$

Applications:

- Operational switching
- Expansion planning: New-build and reconductoring
- Siting: Data centers and storage

Our approach: Randomized switching

- **Relax:** $s \in [0,1]^m$ (switching probabilities)
  - Works for **adding** or **closing** lines
- **Optimize:** via the Frank-Wolfe method (iterative gradient-based optimization):

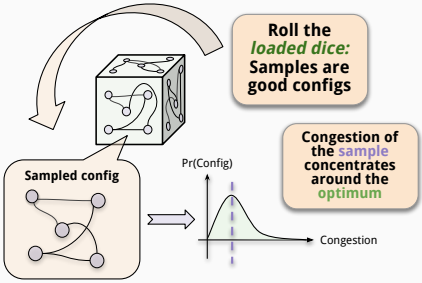
$$(\nabla\varphi(s))_e = -w_e \left|\langle a_e, L_s^+ p\rangle\right|^2 \leq 0$$

- **Sample:** $\tilde{s} \in \{0,1\}^m$ via Bernoulli rounding

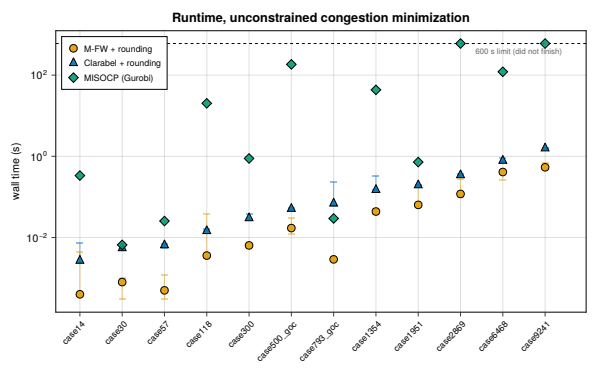
Main Result (Matrix Bernstein)

w.h.p., $\varphi(\tilde{s}) \leq \frac{1+\alpha}{1-\epsilon}\varphi(s_\star)$,
where $s_\star \leftarrow$ optimal probabilities

Families of near-optimal solutions.

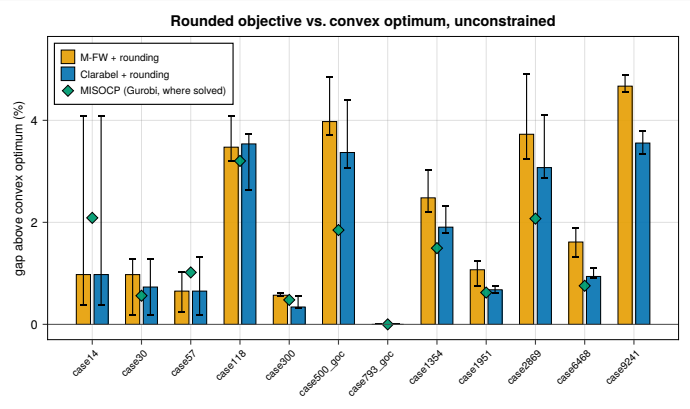


Randomized switching is 10–100× faster than the exact baseline



- 10–100× faster than Gurobi MISOCP, fastest at **every** network scale
- MISOCP **times out** (600 s) on the dense PEGASE cases; Frank–Wolfe finishes in under a second

Median rounded topology lands within 5% of the convex optimum



- Median end-to-end gap (residual fractional suboptimality **plus** rounding loss) stays below 5% of the convex optimum on every RTE/PEGASE case

Expansion planning generalizes switching with an investment cost

Expansion Planning

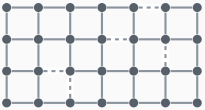
$$\min_{\boldsymbol{s}\in[0,1]^K,\ \|\boldsymbol{s}\|_1\le q} \kappa \mathbf{1}^\mathsf{T}\boldsymbol{s} + \varphi(\boldsymbol{s}), \qquad \varphi(\boldsymbol{s}) = \boldsymbol{p}^\mathsf{T} L_{\boldsymbol{s}}^{+}\boldsymbol{p}, \qquad L_{\boldsymbol{s}} = L_0 + \sum_{k=1}^{K} s_k w_k \boldsymbol{a}_k \boldsymbol{a}_k^\mathsf{T}$$

- L_0: the Laplacian of the existing network. s_k: the built fraction of candidate line k.
- κ: build cost per line. With $\kappa = 0$, building is free.

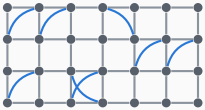
Garver TPAS'70; Ghosh, Boyd, Saberi SIREV'08; **Talkington**, Ostrovskii, Molzahn arXiv'25.

Reconfiguration opens lines; expansion planning adds them

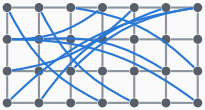
open lines



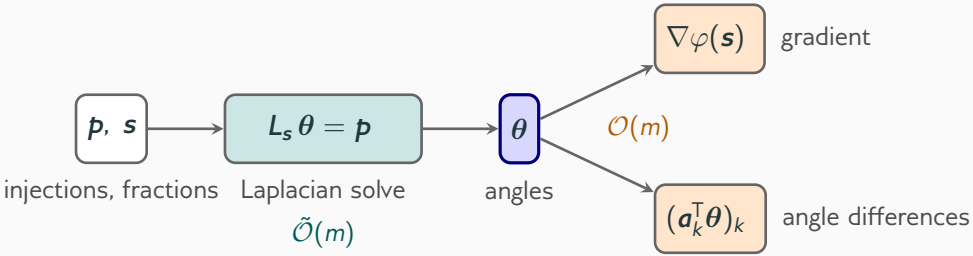
new local lines



new random lines



Each iteration uses one Laplacian solve



Frank-Wolfe for expansion planning

$\boldsymbol{s} \leftarrow \mathbf{0}$

for $t = 0, 1, \ldots$ **do**

- $\boldsymbol{\theta} \leftarrow \boldsymbol{L}_{\boldsymbol{s}}^{+} \boldsymbol{p}$ ▷ one Laplacian solve
- $h_k \leftarrow \kappa - w_k (\boldsymbol{a}_k^{\mathsf{T}} \boldsymbol{\theta})^2$ for each candidate k ▷ the gradient
- $\boldsymbol{z} \leftarrow$ indicator of the q most negative entries of $\boldsymbol{h}$ (negative only) ▷ linear minimization
- $\boldsymbol{s} \leftarrow \boldsymbol{s} + \frac{2}{t+2}(\boldsymbol{z} - \boldsymbol{s})$

Then round $\tilde{s}_k \sim \mathrm{Ber}(s_k)$ independently.

Frank, Wolfe NRLQ'56.

Iteratively reweighted least squares (IRLS): alternate flows and weights

$\varphi(\boldsymbol{s}) = \min_{\boldsymbol{f} \text{ routes } \boldsymbol{p}} \sum_e f_e^2/w_e + \sum_k f_k^2/(s_k w_k)$ over existing lines e and candidates k, minimized by the flow $f_k = s_k w_k \, \boldsymbol{a}_k^\top \boldsymbol{\theta}$, with $\boldsymbol{\theta} = \boldsymbol{L}_{\boldsymbol{s}}^{+} \boldsymbol{p}$.

$\boldsymbol{s} \leftarrow (q/K)\mathbf{1}$ ▷ the budget spread evenly

for $t = 0, 1, \ldots$ **do**

- $\boldsymbol{\theta} \leftarrow \boldsymbol{L}_{\boldsymbol{s}}^{+} \boldsymbol{p}$ ▷ best $\boldsymbol{f}$ for fixed $\boldsymbol{s}$: one Laplacian solve
- $f_k \leftarrow s_k w_k \, \boldsymbol{a}_k^\top \boldsymbol{\theta}$ for each candidate k
- $s_k \leftarrow \min\{1, \ \max\{\delta, \ |f_k|/\sqrt{w_k(\kappa + \lambda)}\}\}$ ▷ best $\boldsymbol{s}$ for fixed $\boldsymbol{f}$

λ: the smallest $\lambda \geq 0$ for which the new $\boldsymbol{s}$ satisfies $||\boldsymbol{s}||_1 \leq q$. $\delta = 10^{-6}$: a floor on s_k.

Suggested by R. Kyng. Straszak, Vishnoi ITCS'17; Adil, Peng, Sachdeva NeurIPS'19; Ene, Vladu ICML'19; Ene, Nguyen, Vladu ICLR'26; Beck SIOPT'15.

A fixed point of IRLS is optimal

1. Building more of line k changes the objective at the rate $\kappa - w_k(\boldsymbol{a}_k^\top \boldsymbol{\theta})^2$.

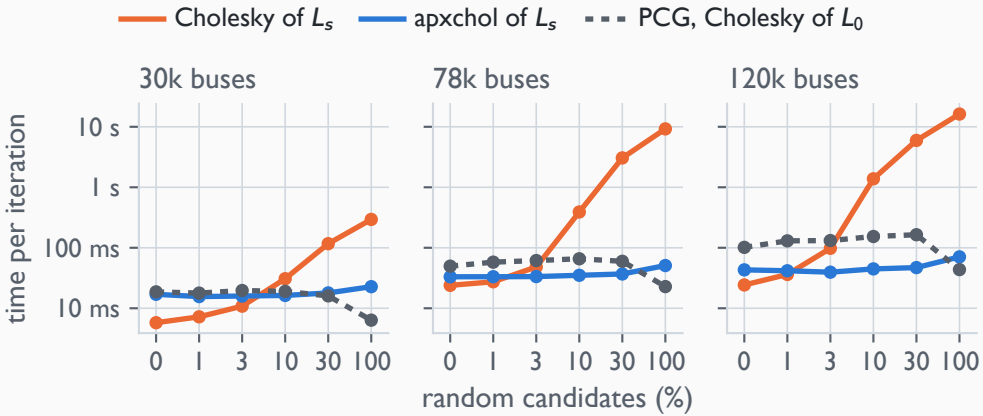
2. The problem is convex, so $\boldsymbol{s}$ is optimal exactly when some $\lambda \geq 0$, zero unless $||\boldsymbol{s}||_1 = q$, gives

$$\kappa - w_k(\boldsymbol{a}_k^\top \boldsymbol{\theta})^2 \quad \begin{cases} = -\lambda & \text{if } \delta < s_k < 1, \\ \leq -\lambda & \text{if } s_k = 1, \\ \geq -\lambda & \text{if } s_k = \delta. \end{cases}$$

   These are the KKT conditions.

3. A fixed point has $s_k = |f_k| / \sqrt{w_k(\kappa + \lambda)}$ with $f_k = s_k w_k \, \boldsymbol{a}_k^\top \boldsymbol{\theta}$, clipped to $[\delta, 1]$. Solving for the rate gives exactly these three cases.

Expansion planning: time per IRLS iteration



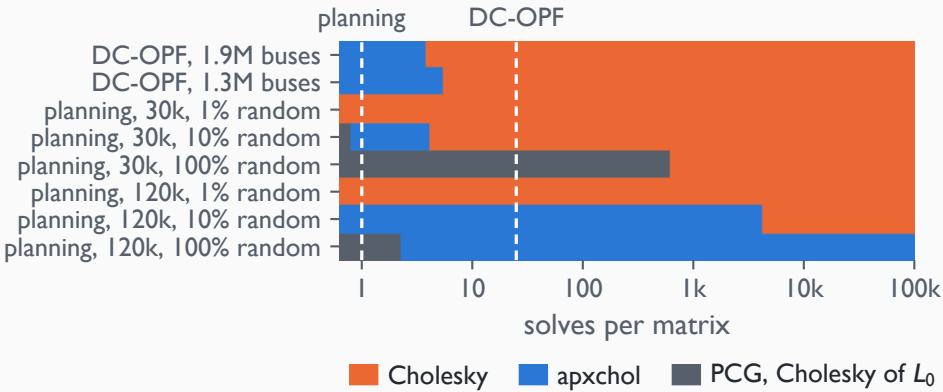
Candidates join random node pairs (horizontal axis) or nodes 2 to 4 lines apart.

Expansion planning with 10% random candidates

	time per iteration (ms)			
nodes	Cholesky of L_s	apxchol of L_s	PCG, Cholesky of L_0	speedup
30k	31	**16**	19	1.2×
78k	390	**35**	66	1.9×
120k	1381	**45**	154	3.4×

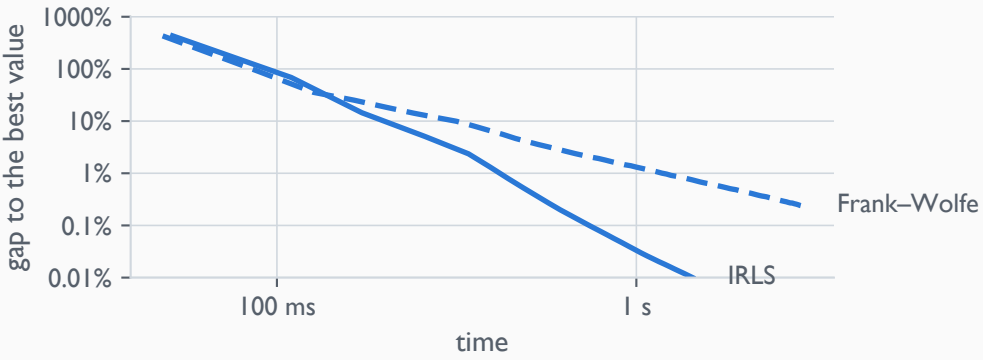
Speedup: apxchol against the next fastest.

The cheapest method depends on the number of solves per matrix



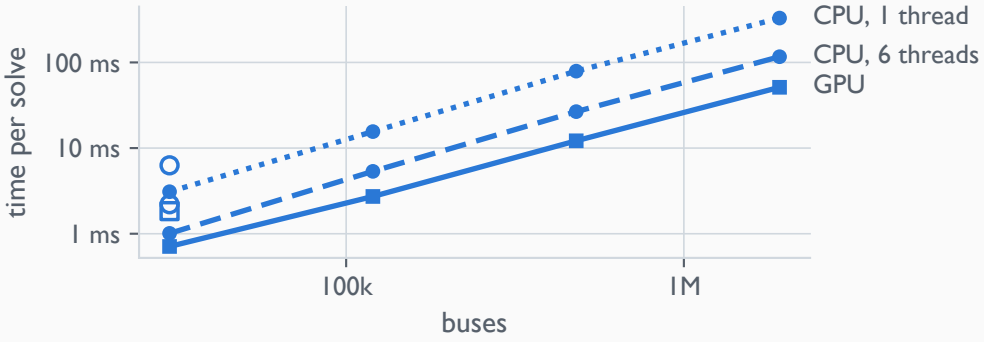
Dashed lines: solves per matrix in planning and in DC-OPF.

IRLS gets within 1% of the best value 2.8× sooner than Frank–Wolfe



120k nodes, 10% random candidates.

PCG with apxchol's factor on the Apple GPU: 1.4 to 2.3× faster than 6 CPU threads



Hollow markers: an expansion planning matrix.

Summary

- DC-OPF: one fixed, sparse Laplacian. Exact Cholesky is faster.
- Expansion planning: a new Laplacian every iteration. With 10% random candidate lines, apxchol is faster: 1.2× at 30k to 3.4× at 120k nodes.
- IRLS reaches a 1% gap 2.8× sooner than Frank–Wolfe at 120k nodes.

Next

- IRLS with randomized rounding for switching.
- For apxchol: many right-hand sides at once, and a factor reused across iterations.
- Complex Laplacians: the admittance matrix $\boldsymbol{Y} = \boldsymbol{A}\,\mathrm{diag}(\boldsymbol{y})\boldsymbol{A}^{\mathsf{T}}$, $\boldsymbol{y} \in \mathbb{C}^m$, is complex symmetric, not Hermitian.
  - Solve $\boldsymbol{Y}\boldsymbol{u} = \boldsymbol{b}$ in $\tilde{\mathcal{O}}(m)$ time?
  - Sparsify $\boldsymbol{Y}$ by sampling line e with probability $\propto |y_e|$ times its effective impedance?

Thank you! Questions?

samuel_talkington@seas.harvard.edu

samueltalkington.com

