Solving security-constrained ACOPF problems

Daniel Bienstock, Columbia University and Gurobi
Richard Waltz, Artelys

A formal textbook statement of standard ACOPF

Minimize cost of generation: $\sum_{g \in \mathcal{G}} F_g(P^g)$

- Here, $\mathcal{G}$ is the set of generators
- P^g is the (active) power generated at g
- F_g is generation cost at g – convex, piecewise-linear or quadratic
  Example: $F_g(P) = 3P^2 + 2P$

Constraints:

- PF (power flow) constraints: choose voltages so that network delivers power from generators to the loads

- Voltage magnitudes are constrained

- Power flow on any line km cannot be too large: $|S_{km}|$ is within limits

- The output of any generator is limited

Minimize $\sum_{g \in \mathcal{G}} F_g(\boldsymbol{P^g})$

with constraints:

$$\boldsymbol{S_{km}} = (G_{kk} - jB_{kk}) \, |\boldsymbol{V_k}|^2 + (G_{km} - jB_{km}) \, |\boldsymbol{V_k}||\boldsymbol{V_m}|(\cos \boldsymbol{\theta_{km}} + \boldsymbol{j} \sin \boldsymbol{\theta_{km}})$$

$$\sum_{km \in \delta(k)} \boldsymbol{S_{km}} = \left(\sum_{\boldsymbol{g \in G(k)}} \boldsymbol{P^g} - P_k^d \right) + j\left(\sum_{\boldsymbol{g \in G(k)}} \boldsymbol{Q^g} - Q_k^d \right)$$

Power flow limit on line km:

$$|\boldsymbol{S_{km}}|^2 = \Re\mathrm{e}(\boldsymbol{S_{km}})^2 + Im(\boldsymbol{S_{km}})^2 \leq U_{km}$$

Voltage limit on node k:

$$V_k^{\min} \leq |\boldsymbol{V_k}| \leq V_k^{\max}$$

Generator output limit on node k:

$$P_k^{\min} \leq \boldsymbol{P_k^g} \leq P_k^{\max}$$

Phase angle limit on line km: (sometimes)

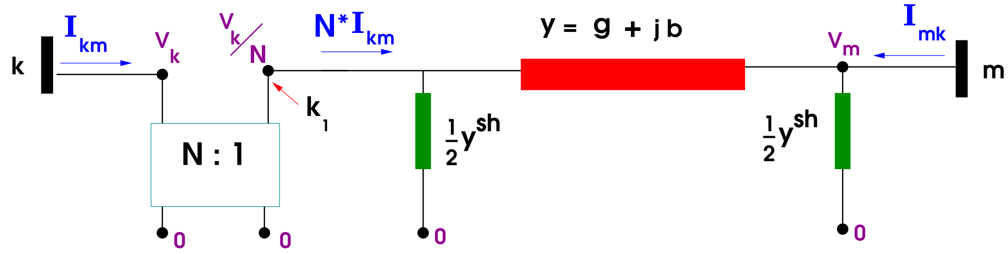
$$|\boldsymbol{\theta_k} - \boldsymbol{\theta_m}| \leq \theta_{k,m}^{\max}$$

GO competition: basic features

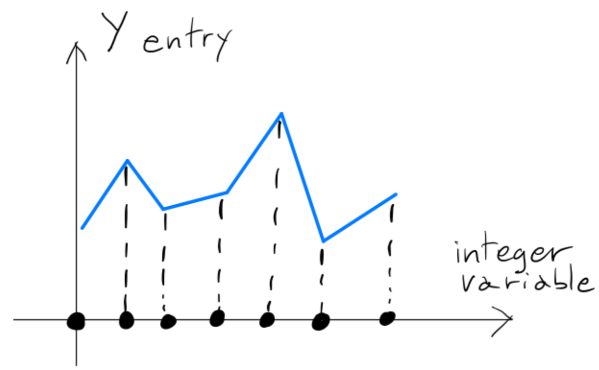
- **Maximize** surplus rather than minimize cost
  - Piecewise-linear function measuring benefit of consumption (for each load)
  - Imbalance, line overload, transformer overload penalty functions (more below)
- Switched shunt step selection
- Tap ratio, phase shift, impedance correction for transformers
- Topology optimization
- Generator commitment, start-up and shut-down decisions
- Security-constrained: total objective is sum of base case plus average contingency case
- More

GO competition: configurable transformers and shunts

- Example: tap ratio and angle in a transformer can be adjusted



- Impedance correction factors modeled using a piecewise-linear curve



- Function can be quite nonlinear with local optima

- Switched shunts, in blocks (at buses)

- Altogether, a large number of integer variables

GO competition: contingencies

- We are given a (often, **large**) list of $\boldsymbol{K}$ *contingencies*
  - "Line out" contingency – one power line fails
  - "Gen out" contingency – one generator
- For each contingency $\boldsymbol{k}$, a feasible solution $\boldsymbol{V^k, G^k, L^k, Z^k}$ must be computed
- The system should be able to migrate from base solution to the contingency solution
- Again, with restrictions: e.g. the difference between base $\boldsymbol{G^b}$ and contingency $\boldsymbol{G^k}$ is constrained
- Total cost =

$$\textbf{cost of base solution} \;+\; \frac{1}{K}\sum \textbf{cost of solution to contingency k}$$

GO competition: handling equation mismatches

- ACOPF model abounds with nonlinear **equations**. Example:

$$\sum_{km \in \delta(k)} \boldsymbol{V_k I^*_{km}} \ = \ \left(\sum_{\boldsymbol{g \in G(k)}} \boldsymbol{P^g} - P^d_k \right) + j\left(\sum_{\boldsymbol{g \in G(k)}} \boldsymbol{Q^g} - Q^d_k \right)$$

- **Challenge!** When presented with a nonlinear equation,

$$\boldsymbol{F(x)} \ = \ \boldsymbol{0},$$

  a numerical solver will instead return $\hat{x}$ with $\boldsymbol{F(\hat{x})} \ = \ \boldsymbol{\epsilon}$
  with $|\boldsymbol{\epsilon}|$ small (usually)

- And then, what?
  A little infeasibility can buy you a lot of optimality

- **GO competition** Add to the objective a term of the form $\boldsymbol{L(|\epsilon|)}$ where $\boldsymbol{L}$ is rapidly increasing convex (infeasibility penalization)

- This difficulty is of a **fundamental** nature!

GO competition: Prior Solution

- $\boldsymbol{V}^{\text{prev}}, \boldsymbol{G}^{\text{prev}}, \boldsymbol{L}^{\text{prev}}, \boldsymbol{Z}^{\text{prev}}$ = existing solution (voltage, generation, controllable loads, configuration)

- To compute: $\boldsymbol{V^b}, \boldsymbol{G^b}, \boldsymbol{L^b}, \boldsymbol{Z^b}$ = new (base) solution

- Why a new solution? Because e.g. some of the fixed loads may change. And the network may be slightly different.

- Restrictions: e.g., the difference between $\boldsymbol{G}^{\text{prev}}$ and $\boldsymbol{G^b}$ is constrained

$$| \boldsymbol{P^{g,prev}} - \boldsymbol{P^{g,b}} | \leq \text{ upper bound dependent on } g,$$

for each generator g.

$\rightarrow$ The prior solution is more help than hindrance!

GO competition: data sets

- Combination of industry and realistic synthetic data sets
- Both large and with many contingencies
- Industry example **C2T2N34363**:
  34,000 buses, **41,000** lines, **900** generators, **3000** contingencies
  thousands of integer variables
- A single ACOPF run on such networks is nontrivial: Example **C2FEN19402**:
  - 19,402 buses, 968 generators, 13,000 lines
  - 267904 variables, 200890 constraints, 6692 contingencies
  - **KNITRO solves Base case** in **51.05** seconds on **11** cores.
- Available time is limited: 5 minutes to one hour

The king of the hill – log barrier methods

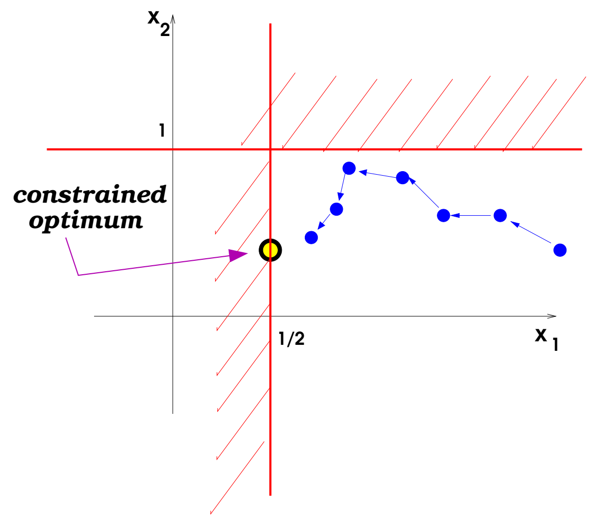
Example: $\min\{\, g(v_1, v_2) \;:\; 1/2 \leq v_1$ and $v_2 \leq 1\}$ where g is convex

becomes:

$$\min \quad g(v_1, v_2) \;-\; \alpha \log(v_1 - 1/2) \;-\; \alpha \log(1 - v_2)$$

subject to: v_1, v_2 **unconstrained**

$\alpha > 0$ a "barrier" parameter (small!)



The king of the hill – log barrier methods

Minimize $f(x)$

s.t. $c(x) = 0$

$x \geq 0$

Instead, solve barrier problems:

Minimize $f(x) \;-\; \mu \sum_{j=1}^{n} \ln(x_j)$

s.t. $c(x) = 0$

And let $\boldsymbol{\mu} \longrightarrow \mathbf{0}$. Primal-dual equations:

$$
\begin{aligned}
\nabla f(x) \;-\; \mu \nabla c(x) \;-\; z &= 0 \\
c(x) &= 0 \\
z_j x_j &= \mu \quad \text{all } j
\end{aligned}
$$

Solve this system, for each fixed μ, using Newton's method

The king of the hill – log barrier methods

Today, two implementations dominate:

- **KNITRO** (Waltz, Nocedal, 2003): A "merit function" method.
- **IPOPT** (Wachter and Biegler, 2004): A "filter" method.

KNITRO and IPOPT followed a long line of work due to many authors!

1. Log barrier methods can (very closely) optimize very large ACOPF problems in **minutes**
2. Nothing else comes close. Relaxation methods only prove bounds – don't provide **solutions**
3. Relaxation methods: spatial branch-and-cut, semidefinite programming, others.

The king of the hill – log barrier methods

Today, two implementations dominate:

- **KNITRO** (Waltz, Nocedal, 2003): A "merit function" method.
- **IPOPT** (Wachter and Biegler, 2004): A "filter" method.

KNITRO and IPOPT followed a long line of work due to many authors!

1. Log barrier methods can (very closely) optimize very large ACOPF problems in **minutes**
2. Nothing else comes close. Relaxation methods only prove bounds – don't provide **solutions**
3. New kid on the block: **Gurobi**. Integrated log-barrier, integer programming and relaxations!

We were # 2 overall

Basic approach:

1. **Dimensionality reduction**
   - Fact: real-world power systems are **very robust**
   - Transition from previous solution, to base solution **should require minor adjustments**
   - Transition from base solution, to contingency solutions **should require minor adjustments**
2. **Progressive rounding**
   Common-sense iterative rounding of integer variables –
   all integer variables modeled as sum of binaries
3. **Numerical stability**
   Some numerical parameters in GO setup are too large.
   Slack penalty function in particular. Replaced with simple linear penalty plus bounds.

Dimensionality reduction

Example: transition from previous solution, to base solution

- Most changes in **data** from previous network, to new one, involve changes in **demands**
- Most demand changes are **very small**
- **What we did:**

1. Select a subset $\boldsymbol{S}$ of buses with **large enough** demand changes

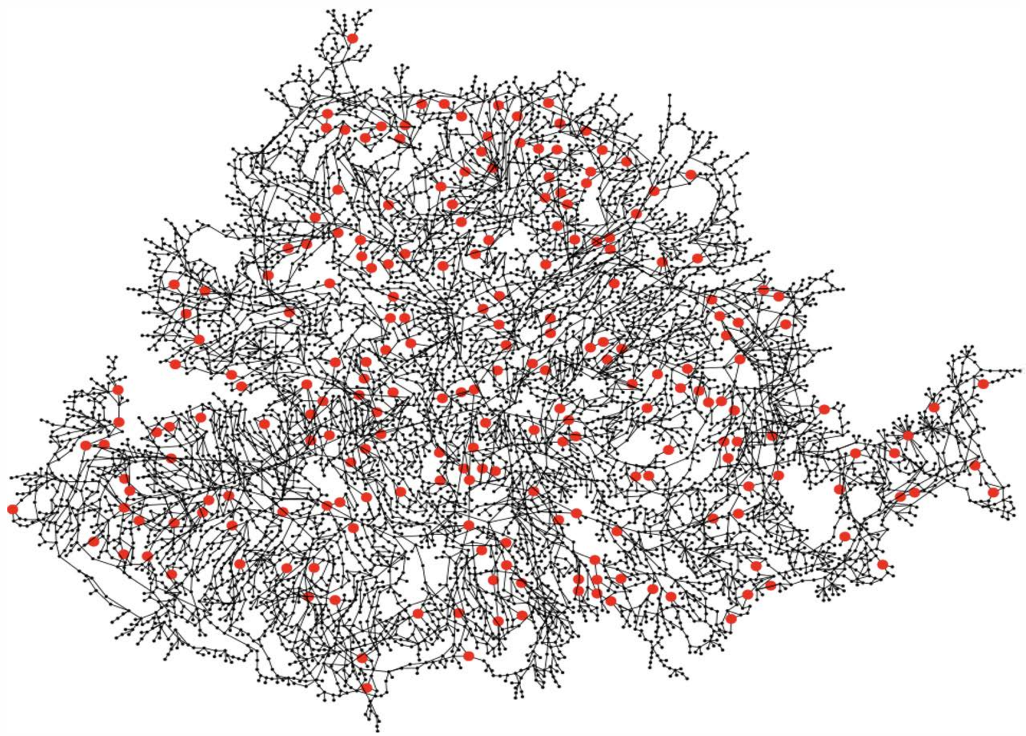
2. **Restrict** changes in prior solution to buses that are **electrically close enough** to $\boldsymbol{S}$

3. Apply our rounding heuristic to the correspondingly restricted ACOPF problem

C2T3N06100 scenario 115

6476 buses, 3371 loads, 406 generators, 5337 lines, 3086 transformers, 2467 contingencies

- About 100,000 variables and constraints
- Picture shows buses where prior solution has infeasibility greater than 1e-3: about 200



obj=712281.64

total_bus_cost 1.85847217e-02

total_load_benefit 1.26257799e+06

total_gen_cost 5.50296330e+05

total_line_cost 0.00000000e+00

total_xfmr_cost 0.00000000e+00

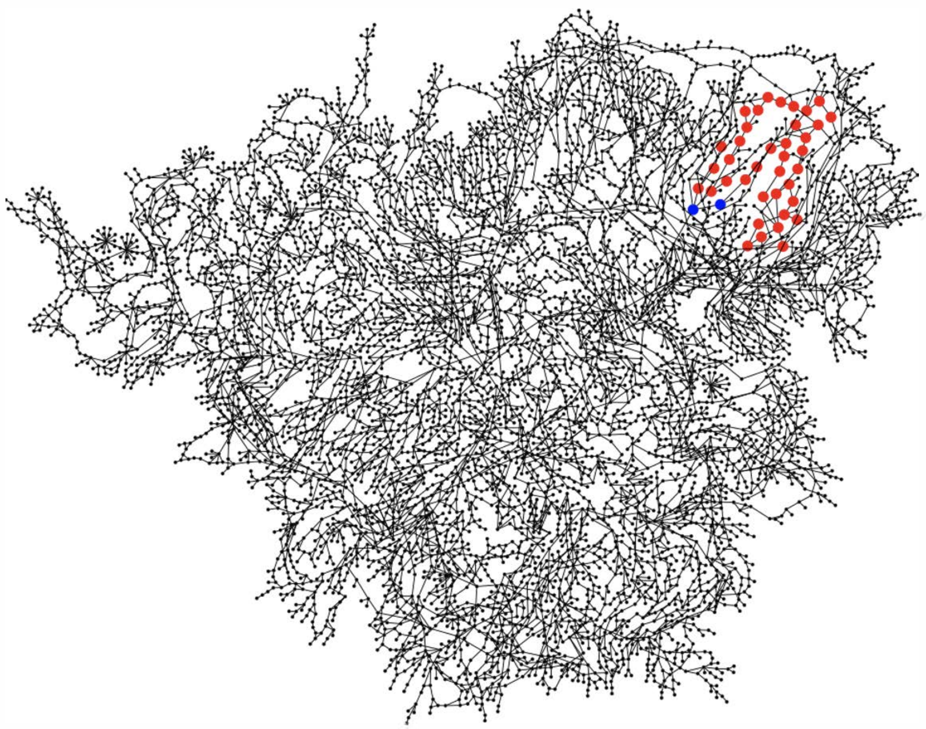
(Knitro feaserror 2.481e-09)

169.05 sec 450 iterations

C2T3N06100 scenario 115

6476 buses, 3371 loads, 406 generators, 5337 lines, 3086 transformers, 2467 contingencies

Contingency 3 (line down)



- **Base case** obj: **712281.64**
- **Heuristic** solution:

total_bus_cost 1.65362151e-02
total_load_benefit 1.25961077e+06
total_gen_cost 5.44767392e+05
total_line_cost 0.00000000e+00
total_xfmr_cost 0.00000000e+00

objective 714843.36

31.30 seconds 227 iterations
+ 6.8 seconds + 27 iterations (rounding)

Algorithm outline

1. Find **good** solution to base case by migrating from prior solution

2. For each contingency, migrate from the base case solution

3. Migration = use the reduced dimensionality heuristic

4. Appropriately handle binary variables (more, below).

5. MPI implementation to four nodes (each node = 16 cores)

Binary variables

1. MINLP branch-and-bound code in Knitro was **not** used – problems too large in GO2.
2. Those binary variables not fixed by dimensionality reduction are addressed by progressively relaxing, rounding, and resolving.
3. Final solve with all integer variables fixed.
4. Knitro solves **warm-started**.
5. Occasionally this procedure fails – in that case our algorithm retreats, unfixes some of the binary variables, and re-runs Knitro.

Basic Details

1. Implementation in **Python**. Why?
   - Incorporate evaluation code into our code.
   - We would have done it in C, but not enough time.
2. We used AMPL to interface with Knitro. Why?
   - Good preprocessor.
   - **Efficient automatic differentiation**.
     The Achylles' heel of log barrier methods:
       compute Hessian of constraints.
3. Not enough time to do as much as we wanted.
   - Last scoring event overlapped with start of bad semester.
   - I introduced a low-level bug within a one-line command.

Datasets in Final Event: 22 networks

- Buses: 403 - 31,777
- Contingencies: 54 - 1800

PRIZES

For Challenge 2, the ARPA-E Benchmark team is not prize eligible and does not occupy a rank during the consideration of prize awards.

Total Prizes ($k) to be Awarded, Subject to Eligibility

Team	Trial Event 3	Final Event	FE + T3
GravityX	130	600	730
Artelys	170	360	530
GOT-BSI-OPF	0	420	420
Pearl Street Technologies	70	270	340
Electric Stampede	140	0	140
GMI-GO	60	60	120
Monday Mornings	0	60	60
GO-SNIP	0	30	30
Gordian Knot	30	0	30
total	600	1,800	2,400