Problems and solutions in nonlinear mixed-integer programming

Daniel Bienstock
Columbia University

IMA, 2016
Talk outline

1. Some light entertainment

2. Some mathematics

3. Additional entertainment
Why we should study polynomial optimization:

**cascading failures of power grids**

• In August 2003, a cascading failure of the Eastern Interconnect caused a large and long-lasting blackout

• The Eastern Interconnect is the electrical circuit that we are in

• The blackout affected some fifty million people for several days and cost a lot of money

• In September, 2003, a similar blackout affected most of Italy
Recent cascades

- U.S. Northeast and Canada; Italy, 2003
- San Diego, 2011
- India, 2012

Rising concerns

- Increasing demand, increasing scope and complexity of grids
- Too expensive to add extensive capacity
- Use of renewables desirable but adds stochastic risk
- Malevolent action (?)
Cascade dynamics

(0) Stuff happens (“act of God”): some network elements (power lines, generators, transformers, etc) are disabled
Cascade dynamics

(0) Stuff happens (“act of God”): some network elements (power lines, generators, transformers, etc) are disabled

(1) Power flows are rearranged: mostly due to physics
Cascade dynamics

(0) Stuff happens (“act of God”): some network elements
   (power lines, generators, transformers, etc)
   are disabled

(1) Power flows are rearranged: mostly due to physics

(2) As a result some network elements become overloaded
Cascade dynamics

(0) Stuff happens ("act of God"): some network elements

   (power lines, generators, transformers, etc)

   are **disabled**

(1) Power flows are rearranged: mostly due to physics

(2) As a result some network elements become overloaded

(3) At a later time, some of these become tripped or outaged
Cascade dynamics

(0) Stuff happens ("act of God"): some network elements (power lines, generators, transformers, etc) are disabled

(1) Power flows are rearranged: mostly due to physics

(2) As a result some network elements become overloaded

(3) At a later time, some of these become tripped or outaged

(4) Go to (1).
Let’s go to the movies
Simulated cascade of Eastern Interconnect
Simulated cascade of Eastern Interconnect
Simulated cascade of Eastern Interconnect
Simulated cascade of Eastern Interconnect
Simulated cascade of Eastern Interconnect
Simulated cascade of Eastern Interconnect
Simulated cascade of Eastern Interconnect
Simulated cascade of Eastern Interconnect
Simulated cascade of Eastern Interconnect
Simulated cascade of Eastern Interconnect
Simulated cascade of Eastern Interconnect
Simulated cascade of Eastern Interconnect
Simulated cascade of Eastern Interconnect
Simulated cascade of Eastern Interconnect
Simulated cascade of Eastern Interconnect
Back to reality:

why it is hard to simulate a power grid under distress

(1) We have to explain when and why equipment will fail

(2) This requires an understanding of the physics of power flows

(3) Additionally, there is noise, missing information, and more

→ let’s begin with (2).
The Grid

Conductor

Steam magnetic field

Stator-rotor

Source

Energy

Current, voltage

$\omega$
Voltage, Power, Current

**Real-time voltage** (potential energy) at bus (node) $k$:

$$V_k(t) = \hat{V}_k \cos(\omega t + \theta_k)$$

**Steady-state** (time average over one period of length $2\pi/\omega$): voltage at bus $k$ represented as:

$$\hat{V}_ke^{j\theta_k} = \hat{V}_k(\cos \theta_k + j \sin \theta_k)$$

- $I_{km} = (\text{complex}) \text{ current}$ injected into $km$ at $k$
- $S_{km} = (\text{complex}) \text{ power}$ injected into $km$ at $k$
Voltage, Power, Current

Real-time **voltage** (potential energy) at bus (node) \( k \):

\[
V_k(t) = \hat{V}_k \cos(\omega t + \theta_k)
\]

Steady-state (time average over one period of length \( 2\pi/\omega \)):

\[
\text{voltage at bus } k \text{ represented as: } \hat{V}_k e^{j\theta_k} = \hat{V}_k (\cos \theta_k + j \sin \theta_k)
\]

- \( I_{km} \) = (complex) **current** injected into \( km \) at \( k \)
- \( S_{km} \) = (complex) **power** injected into \( km \) at \( k \)
- Ohm’s law: \( I_{km} = y_{km}(V_k - V_m) \) (\( y_{km} \) = admittance)
- \( S_{km} = V_k I_{km}^* \)
Steady-state (time average over one period of length $2\pi/\omega$): voltage at bus $k$ represented as:

$$= \hat{V}_k e^{j\theta_k} = \hat{V}_k (\cos \theta_k + j \sin \theta_k)$$

- $I_{km} =$ (complex) current injected into $km$ at $k$
- $S_{km} =$ (complex) power injected into $km$ at $k$
- Ohm’s law: $I_{km} = y_{km} (V_k - V_m)$ ($y_{km} =$ admittance)
- $S_{km} = V_k I_{km}^*$

These are all complex quantities, but all are “real”
Steady-state (time average over one period of length $2\pi/\omega$): voltage at bus $k$ represented as:

$$V_k = \hat{V}_k e^{j\theta_k} = \hat{V}_k (\cos \theta_k + j \sin \theta_k)$$

- $I_{km} = \text{(complex) current}$ injected into $km$ at $k$
- $S_{km} = \text{(complex) power}$ injected into $km$ at $k$
- Ohm’s law: $I_{km} = y_{km} (V_k - V_m)$ ($y_{km} = \text{admittance}$)
- $S_{km} = V_k I_{km}^*$

These are all complex quantities, but all are “real”

- Real part of $S_{km} = P_{km} = \text{“active” power}$
- Imaginary part of $S_{km} = Q_{km} = \text{“reactive” power}$
Steady-state (time average over one period of length $2\pi/\omega$): voltage at bus $k$ represented as:

$$V_k e^{j\theta_k} = \hat{V}_k (\cos \theta_k + j \sin \theta_k)$$

- $I_{km}$ = (complex) current injected into $km$ at $k$
- $S_{km}$ = (complex) power injected into $km$ at $k$
- Ohm’s law: $I_{km} = y_{km} (V_k - V_m)$ ($y_{km} =$ admittance)
- $S_{km} = V_k I_{km}^*$

These are all complex quantities, but all are “real”

- Real part of $S_{km} = P_{km}$ = “active” power
- Imaginary part of $S_{km} = Q_{km}$ = “reactive” power

If we write $V_k = e_k + j f_k$, then

$$P_{km} = (e_k - e_m)(g, b)(e_k^{f_k}) + (f_k - f_m)(-b, g)(e_k^{f_k})$$

(Here, $y_{km} = g + jb$, a quadratic expression on $e_k, e_m, f_k, f_m$.
- A similar quadratic yields $Q_{km}$
Steady-state (time average over one period of length $2\pi/\omega$): voltage at bus $k$ represented as:

$$\hat{V}_k e^{j\theta_k} = \hat{V}_k (\cos \theta_k + j \sin \theta_k)$$

- $I_{km} =$ (complex) current injected into $km$ at $k$
- $S_{km} =$ (complex) power injected into $km$ at $k$
- Ohm’s law: $I_{km} = y_{km}(V_k - V_m)$ ($y_{km} =$ admittance)
- $S_{km} = V_k I_{km}^*$

These are all complex quantities, but all are “real”

- **Real** part of $S_{km} = P_{km} =$ “active” power
- **Imaginary** part of $S_{km} = Q_{km} =$ “reactive” power
- If we write $V_k = e_k + jf_k$, then

$$P_{km} = (e_k - e_m)(g, b)(e_k^{f_k}) + (f_k - f_m)(-b, g)(e_k^{f_k}).$$

(Here, $y_{km} = g + jb$, a quadratic expression on $e_k, e_m, f_k, f_m$.

- A similar quadratic yields $Q_{km}$
- What do we have at a given bus $k$?

**total power injected by k** = injection into k1 + injection into k2 + injection into k3
Putting it all together: power flow problem

\[ V_k = \hat{V}_ke^{j\theta_k} = e_k + jf_k, \]  \hspace{1cm} \text{(1)}

\[ I_{km} = y_{k,m}(V_k - V_m), \quad y_{k,m} = \text{admittance of km}. \] \hspace{1cm} \text{(2)}

\[ p_{km} = \Re(V_kI_{km}^*), \quad q_{km} = \Im(V_{km}I_{km}^*) \] \hspace{1cm} \text{(3)}

Network Equations

\[ \sum_{km \in \delta(k)} p_{km} = \hat{P}_k, \quad \sum_{km \in \delta(k)} q_{km} = \hat{Q}_k \quad \forall \ k \] \hspace{1cm} \text{(4)}

**Generator:** \( \hat{P}_k, |V_k| (= \hat{V}_k) \) given. Other buses: \( \hat{P}_k, \hat{Q}_k \) given.

**Problem.** Compute a solution of this system of quadratic equations.
More general problem: ACOPF

\[ V_k = \hat{V}_k e^{j\theta_k^V} = e_k + jf_k, \]  

(5)

\[ I_{km} = y_{k,m}(V_k - V_m), \ y_{k,m} = \text{admittance of } km. \]  

(6)

\[ p_{km} = \text{Re}(V_k I_{km}^*), \ q_{km} = \text{Im}(V_{km} I_{km}^*) \]  

(7)

Network Inequalities

\[ \hat{P}_k \text{min} \leq \sum_{km \in \delta(k)} p_{km} \leq \hat{P}_k \text{max}, \ \hat{Q}_k \text{min} \leq \sum_{km \in \delta(k)} q_{km} \leq \hat{Q}_k \text{max} \ \forall \ k \]  

(8)

\[ \hat{V}_k \text{min} \leq |V_k| \leq \hat{V}_k \text{max} \ \forall \ k \]  

(9)

Problem

Solve an optimization problem subject to these quadratic inequalities.
How is ACOPF solved in industrial practice?

• Best practice #1:
How is ACOPF solved in industrial practice?

- **Best practice #1**: Don’t solve it and go for a beer instead

Solve a *linearized* version.

**Why?**

**Should be that:** \(|V_k| \approx 1\) for all \(k\), so assume \(|V_k| = 1\)

**and:** \(\theta_k \approx \theta_m\), so \(\sin(\theta_k - \theta_m) \to \theta_k - \theta_m\) and \(\cos(\theta_k - \theta_m) \to 1\)
How is ACOPF solved in industrial practice?

- **Best practice #1**: Don’t solve it and go for a beer instead

  Solve a linearized version.

  Why?

  **Should be that:** \(|V_k| \approx 1\) for all \(k\), so assume \(|V_k| = 1\)

  and: \(\theta_k \approx \theta_m\), so \(\sin(\theta_k - \theta_m) \rightarrow \theta_k - \theta_m\) and \(\cos(\theta_k - \theta_m) \rightarrow 1\)

- **Sequential linearization.** Replace all active constraints with their linearizations, and iterate.
How is ACOPF solved in industrial practice?

- **Best practice #1**: Don’t solve it and go for a beer instead

  Solve a linearized version.

  Why?

  **Should be that**: \(|V_k| \approx 1\) for all \(k\), so assume \(|V_k| = 1\) and:
  \[
  \theta_k \approx \theta_m, \quad \sin(\theta_k - \theta_m) \to \theta_k - \theta_m \quad \text{and} \quad \cos(\theta_k - \theta_m) \to 1
  \]

- **Sequential linearization**. Replace all active constraints with their linearizations, and iterate.

- **IPOPT, et al.** Use interior point (e.g. barrier) methods to obtain a locally optimal solution.

  → But can we “certify” optimality?

  → But can we “certify” **infeasibility**?
Quadratically constrained, quadratic programming problems (QCQPs):

\[
\begin{align*}
\min & \quad f_0(x) \\
\text{s.t.} & \quad f_i(x) \leq 0, \quad 1 \leq i \leq m \\
& \quad x \in \mathbb{R}^n
\end{align*}
\]

Here,

\[ f_i(x) = x^T M_i x + c_i^T x + d_i \]

is a general quadratic, with \( M_i \) \( n \times n \), wlog symmetric.
Quadratically constrained, quadratic programming problems (QCQPs):

$$\begin{align*}
\min & \quad f_0(x) \\
\text{s.t.} & \quad f_i(x) \leq 0, \quad 1 \leq i \leq m \\
& \quad x \in \mathbb{R}^n
\end{align*}$$

Here,

$$f_i(x) = x^T M_i x + c_i^T x + d_i$$

is a general quadratic, with $M_i$ $n \times n$, wlog symmetric

- **Special case: Linear Programming**
  $M_i = 0, \quad 0 \leq i \leq m$

- **Special case: Convex Quadratic Programming:**
  $M_i \succeq 0, \quad 0 \leq i \leq m$

- **Unfortunately**, QCQP is NP-hard
Quadratically constrained, quadratic programming problems (QCQPs):

\[
\begin{align*}
\min & \quad f_0(x) \\
\text{s.t.} & \quad f_i(x) \leq 0, \quad 1 \leq i \leq m \\
& \quad x \in \mathbb{R}^n
\end{align*}
\]

Here,

\[ f_i(x) = x^T M_i x + c_i^T x + d_i \]

is a general quadratic, with \( M_i \ n \times n \), wlog symmetric

- **Special case: Linear Programming**
  \( M_i = 0, \ 0 \leq i \leq m \)

- **Special case: Convex Quadratic Programming:**
  \( M_i \succeq 0, \ 0 \leq i \leq m \)

- **Unfortunately**, QCQP is NP-hard

  **a deep fact:** \( x_j(1 - x_j) = 0 \) is a quadratic constraint
OK, let’s take a step waaaaay back: the trust-region (sub)problem

\[
\begin{align*}
\min & \quad x^T Q x + c^T x \\
\text{s.t.} & \quad \| x - \mu \|_2 \leq r
\end{align*}
\]
OK, let’s take a step waaaaay back: the trust-region (sub)problem

\[
\begin{align*}
\min & \quad x^T Q x + c^T x \\
\text{s.t.} & \quad \|x - \mu\|_2 \leq r
\end{align*}
\]

- Control Theory
- Dynamical Systems
- Robust error estimation
- Robust optimization
- Olympic swimming
OK, let’s take a step waaaaay back: the trust-region (sub)problem

\[
\begin{align*}
\min & \quad x^T Q x + c^T x \\
\text{s.t.} & \quad \|x - \mu\|_2 \leq r
\end{align*}
\]

- Control Theory
- Dynamical Systems
- Robust error estimation
- Robust optimization
- ...
- How about the antitrust region (sub)problem

\[
\begin{align*}
\min & \quad x^T Q x + c^T x \\
\text{s.t.} & \quad \|x - \mu\|_2 \geq r
\end{align*}
\]
Digression: application of trust-region subproblem

→ Unconstrained optimization  \( \min\{f(x) : x \in \mathbb{R}^n\} \)

\textbf{Algorithm}

- Given an iterate \( x^t \), construct a \textbf{quadratic} “model” for \( f(x) \) which is approximately valid in a neighborhood \( \|x - x^t\| \leq \Delta \).

- For example, use
  \[
  f(x^k) + \frac{1}{2}(x - x^t)^T H(x^t)(x - x^t)
  \]
  where \( H(x^t) \) is the Hessian of \( f \) at \( x^t \).
Digression: application of trust-region subproblem

Unconstrained optimization $\min\{f(x) : x \in \mathbb{R}^n\}$

Algorithm

• Given an iterate $x^t$, construct a quadratic “model” for $f(x)$ which is approximately valid in a neighborhood $\|x - x^t\| \leq \Delta$.

• For example, get pairs $(y^1, f(y^1)), (y^2, f(y^2)), \ldots, (y^m, f(y^m))$

• Using these samples, construct an approximation to $f(x)$ (model = spline, least squares estimate, etc).

• Call this model: $Q(x)$

• Solve: $\min\{Q(x) : \|x - x^t\| \leq \Delta\}$. This is the trust-region subproblem.

• The solution becomes $w^{t+1}$.
  Or (better): conduct a line-search from $w^t$ to the solution so as to compute $w^{t+1}$.

• General purpose codes: KNITRO, LOQO have been used on OPF.
Summary

→ Unconstrained optimization  \( \min\{ f(x) : x \in \mathbb{R}^n \} \)

Algorithm

• Given an iterate  \( x^t \), construct a \textbf{quadratic} “model” for  \( f(x) \) which is approximately valid in a neighborhood  \( \|x - x^t\| \leq \Delta \).

• Call this model:  \( Q(x) \)

• \textbf{Solve: } \( \min\{ Q(x) : \|x - x^t\| \leq \Delta \} \).
  This is the trust-region subproblem.

• The solution becomes  \( w^{t+1} \).
  Or \textbf{(better): } conduct a line-search from  \( w^t \) to the solution so as to compute  \( w^{t+1} \).

• \textbf{What does this algorithm produce?}

• Does it solve the problem? Approximately?
How do we solve the trust region subproblem?

- Fast solution is crucial for the application
- This is a very mature problem that is considered well-solved
- Let us look at the problem from a broader perspective
Want to solve:

\[ f^* = \min f(x) = x^T Ax + 2a^T x + a_0 \]
\[ \text{s.t. } g(x) = x^T Bx + 2b^T x + b_0 \geq 0 \]
Want to solve:

\[
\begin{align*}
    f^* &= \min f(x) = x^T A x + 2 a^T x + a_0 \\
    \text{s.t.} & \quad g(x) = x^T B x + 2 b^T x + b_0 \geq 0
\end{align*}
\]

Easier question:

Given a real \( \theta \), is it the case that \( f^* \geq \theta \)?
Want to solve:

\[
\mathbf{f}^* = \min \; f(x) = x^T A x + 2 a^T x + a_0
\]

s.t. \quad \mathbf{g}(x) = x^T B x + 2 b^T x + b_0 \geq 0

Easier question:

Given a real \( \theta \), is it the case that \( \mathbf{f}^* \geq \theta \)?

Duality: true, iff there exists real \( \gamma \geq 0 \) s.t. \( f(x) - \theta - \gamma g(x) \geq 0 \; \forall \; x \)

(this is MAGIC)
Want to solve:

\[
f^\ast = \min f(x) \doteq x^T Ax + 2a^T x + a_0
\]

s.t. \( g(x) \doteq x^T Bx + 2b^T x + b_0 \geq 0 \)

**Easier question:**
Given a real \( \theta \), is it the case that \( f^* \geq \theta \)?

**Duality:** true, iff there exists real \( \gamma \geq 0 \) s.t. \( f(x) - \theta - \gamma g(x) \geq 0 \ \forall \ x \)

(i.e., iff) there exists \( \gamma \geq 0 \) with:

\[
(x^T, 1) \begin{pmatrix} A - \gamma B & a - \gamma b \\ (a - \gamma b)^T & a_0 - \gamma b_0 - \theta \end{pmatrix} \begin{pmatrix} x \\ 1 \end{pmatrix} \geq 0 \ \forall \ x
\]
Want to solve:
\[
\begin{align*}
\mathbf{f}^* &= \min f(x) = x^T A x + 2a^T x + a_0 \\
\text{s.t. } g(x) &= x^T B x + 2b^T x + b_0 \geq 0
\end{align*}
\]

**Easier question:**
Given a real \( \theta \), is it the case that \( \mathbf{f}^* \geq \theta \)?

**Duality:** true, iff there exists real \( \gamma \geq 0 \) s.t.
\[
\mathbf{f}(x) - \theta - \gamma g(x) \geq 0 \quad \forall \ x
\]

(i.e., iff) there exists \( \gamma \geq 0 \) with:
\[
(x^T, 1) \begin{pmatrix} A - \gamma B & a - \gamma b \\ (a - \gamma b)^T & a_0 - \gamma b_0 - \theta \end{pmatrix} \begin{pmatrix} x \\ 1 \end{pmatrix} \geq 0 \quad \forall \ x
\]

and it turns out that this is equivalent to:
\[
\begin{pmatrix} A - \gamma B & a - \gamma b \\ (a - \gamma b)^T & a_0 - \gamma b_0 - \theta \end{pmatrix} \succeq 0 \quad \text{(proof?)}
\]
\[ f^* = \min f(x) = x^T Ax + 2a^T x + a_0 \]
\[ \text{s.t. } g(x) = x^T Bx + 2b^T x + b_0 \geq 0 \]

Rewrite it as:

\[ \max \theta \]
\[ \text{s.t. } f^* \geq \theta \]

Duality:

\[ \max_{\theta, \gamma} \theta \]
\[ \text{s.t. } \begin{pmatrix} A - \gamma B & a - \gamma b \\ (a - \gamma b)^T & a_0 - \gamma b_0 - \theta \end{pmatrix} \succeq 0 \]
Back to general QCQP

(QCQP):  \[
\begin{align*}
\text{min} & \quad x^T Q x + 2 c^T x \\
\text{s.t.} & \quad x^T A_i x + 2 b_i^T x + r_i \geq 0 \quad i = 1, \ldots, m \\
x & \in \mathbb{R}^n.
\end{align*}
\]
Back to general QCQP

(QCQP): \[
\begin{align*}
\text{min} & \quad x^T Q x + 2c^T x \\
\text{s.t.} & \quad x^T A_i x + 2b_i^T x + r_i \geq 0 \quad i = 1, \ldots, m \\
& \quad x \in \mathbb{R}^n.
\end{align*}
\]

→ form the semidefinite relaxation

(SR): \[
\begin{align*}
\text{min} & \quad \left( \begin{array}{cc}
0 & c^T \\
c & Q
\end{array} \right) \bullet X \\
\text{s.t.} & \quad \left( \begin{array}{cc}
r_i & b_i^T \\
b_i & A_i
\end{array} \right) \bullet X \geq 0 \quad i = 1, \ldots, m \\
& \quad X \succeq 0, \quad X_{11} = 1.
\end{align*}
\]

Here, for symmetric matrices \( M, \ N \),

\[ M \bullet N = \sum_{h,k} M_{hk} N_{hk} \]

Why do we call it a relaxation?
Back to general QCQP

(QCQP): \[ \begin{align*}
& \text{min } x^T Q x + 2 c^T x \\
& \text{s.t. } x^T A_i x + 2 b_i^T x + r_i \geq 0 \quad i = 1, \ldots, m \\
& x \in \mathbb{R}^n. 
\end{align*} \]

→ form the **semidefinite relaxation**

(SR): \[ \begin{align*}
& \text{min } \begin{pmatrix} 0 & c^T \\ c & Q \end{pmatrix} \begin{pmatrix} x \\ x^T \end{pmatrix} \\
& \text{s.t. } \begin{pmatrix} r_i & b_i^T \\ b_i & A_i \end{pmatrix} \begin{pmatrix} x \\ x^T \end{pmatrix} \geq 0 \quad i = 1, \ldots, m \\
& X \succeq 0, \quad X_{11} = 1. 
\end{align*} \]

Here, for symmetric matrices \( M, N \),

\[ M \bullet N = \sum_{h,k} M_{hk} N_{hk} \]

Why do we call it a relaxation?

Given \( x \) feasible for QCQP, the matrix \( X = (1, x^T) \begin{pmatrix} 1 \\ x \end{pmatrix} \) feasible for SR and with the same value

So the value of problem SR is a **lower bound** for QCQP
Back to general QCQP

(QCQP): $\min \ x^T Q x + 2 c^T x$

s.t. $x^T A_i x + 2 b_i^T x + r_i \geq 0 \quad i = 1, \ldots, m$

$x \in \mathbb{R}^n$.

→ form the semidefinite relaxation

(SR): $\min \ \left( \begin{array}{cc} 0 & c^T \\ c & Q \end{array} \right) \bullet X$

s.t. $\left( \begin{array}{cc} r_i & b_i^T \\ b_i & A_i \end{array} \right) \bullet X \geq 0 \quad i = 1, \ldots, m$

$X \succeq 0, \quad X_{11} = 1$.

Here, for symmetric matrices $M, \ N$,

$M \bullet N = \sum_{h,k} M_{hk} N_{hk}$

Why do we call it a relaxation?

Given $x$ feasible for QCQP, the matrix $X = (1, x^T) \left( \begin{array}{c} 1 \\ x \end{array} \right)$ feasible for SR and with the same value

So the value of problem SR is a lower bound for QCQP

But we need to go backwards: given a solution $X$ to SR, does it give us a solution to QCQP?

Only if $X$ has rank-1. Unfortunately, SR typically does not have a rank-1 solution.
It’s pretty bad ...

Theorem (Pataki, 1998):

An SDP

(SR): \[ \min M \cdot X \]
\[ \text{s.t. } N^i \cdot X \geq b_i, \quad i = 1, \ldots, m \]
\[ X \succeq 0, \quad X \text{ an } n \times n \text{ matrix,} \]

always has a solution of rank \( \approx m^{1/2} \), and this bound is attained.

Observation (Lavaei and Low):
The SDP relaxation of practical AC-OPF instances can have a rank-1 solution, or the solution can be relatively easy to massage into rank-1 solutions (also see earlier work of Bai et al)

**Current research thrust:** Can we leverage this observation into practical, globally optimal algorithms for AC-OPF?
I need to solve a complicated QCQP

(QCQP): \[ \min x^T Q x + 2 c^T x \]
\[ \text{s.t. } x^T A_i x + 2 b_i^T x + r_i \geq 0 \quad i = 1, \ldots, m \]
\[ x \in \mathbb{R}^n. \]

... what do I do?
I need to solve a complicated QCQP

(QCQP): \[
\begin{align*}
\text{min } & \quad x^T Q x + 2 c^T x \\
\text{s.t. } & \quad x^T A_i x + 2 b_i^T x + r_i \geq 0 \quad i = 1, \ldots, m \\
& \quad x \in \mathbb{R}^n.
\end{align*}
\]

... what do I do? run away

General techniques

- McCormick reformulation.
  Each \(x_i x_j\), where \(x_i^L \leq x_i \leq x_i^U\) and \(x_j^L \leq x_j \leq x_j^U\) is replaced by \(X_{ij}\) plus
  \[
  \begin{align*}
  X_{ij} & \geq x_i^L x_j + x_j^L x_i - x_i^L x_j^L \\
  X_{ij} & \geq x_i^U x_j + x_j^U x_i - x_i^U x_j^U \\
  X_{ij} & \leq x_i^U x_j + x_j^L x_i - x_i^U x_j^L \\
  X_{ij} & \leq x_i^L x_j + x_j^U x_i - x_i^L x_j^U
  \end{align*}
  \]

  Yields a linear programming relaxation

- Spatial branching, e.g. if \(0 \leq x_j \leq 1\) you branch as: \(0 \leq x_j \leq 1/2\) and \(1/2 \leq x_j \leq 1\).

- Widely implemented in many high-quality codes.

Let’s take a computing break
A nice generalization of the trust-region subproblem

Solve a problem of the form

\[
\min \quad x^T Q x + c^T x \\
\text{s.t.} \quad \|x\|_2 \leq 1 \\
a_i^T x \leq b_i \quad i = 1, 2
\]

provided the two (two!) linear constraints are parallel:
A nice generalization of the trust-region subproblem

Solve a problem of the form

\[
\min \ x^T Q x + c^T x \\
\text{s.t.} \quad \|x\|_2 \leq 1 \\
a_i^T x \leq b_i \quad i = 1, 2
\]

provided the two (two!) linear constraints are parallel:

\[
\min \ \{ x^T Q x + c^T x : d \leq x_1 \leq u, \|x\| \leq 1 \}
\]

restate as: \[ \min \sum_{i,j} q_{ij} X_{ij} + c^T x \]

\text{s.t.} \quad \begin{align*}
X_{11} + du & \leq (d + u)x_1 \\
\|X_{1} - dx\| & \leq x_1 - d \\
\|ux - X_{1}\| & \leq u - x_1 \\
\sum_{j} X_{jj} & \leq 1
\end{align*} \\
X \succeq xx^T
\]

Lemma: This problem has an optimal solution with \( X = xx^T \), i.e. a rank-1 solution.
Many theoretically nice generalizations

- More than one ball constraint (but not too many) and more than one linear inequality (but not too many)

- A “small” number of general quadratic constraints

- The algorithms are theoretically efficient but computationally very challenging

- I did some of this, so let’s move on
(QCQP): \[ \begin{align*} 
\text{min} & \quad x^T Q x + 2 c^T x \\
\text{s.t.} & \quad x^T A_i x + 2 b_i^T x + r_i \geq 0 \quad i = 1, \ldots, m \\
x & \in \mathbb{R}^n.
\end{align*} \]

→ form the \textbf{semidefinite relaxation}

(SR): \[ \begin{align*} 
\text{min} & \quad \begin{pmatrix} 0 & c^T \\ c & Q \end{pmatrix} \bullet X \\
\text{s.t.} & \quad \begin{pmatrix} r_i & b_i^T \\ b_i & A_i \end{pmatrix} \bullet X \geq 0 \quad i = 1, \ldots, m \\
X & \succeq 0, \quad X_{11} = 1.
\end{align*} \]

And let’s make it worse. How about the \textbf{moment relaxation}?
Higher-order SDP relaxations

Consider the polynomial optimization problem

\[ f_0^* \doteq \min \{ f_0(x) : f_i(x) \geq 0, \ 1 \leq i \leq m, \ x \in \mathbb{R}^n \}, \]

where each \( f_i(x) \) is a polynomial i.e. \( f_i(x) = \sum_{\pi \in S(i)} a_{i,\pi} x^{\pi} \).

- Each \( \pi \) is a tuple \( \pi_1, \pi_2, \ldots, \pi_n \) of nonnegative integers, and \( x^{\pi} \doteq x_1^{\pi_1} x_2^{\pi_2} \cdots x_n^{\pi_n} \).
- Each \( S(i) \) is a finite set of tuples, and the \( a_{i,\pi} \) are reals.
Higher-order SDP relaxations

Consider the polynomial optimization problem

\[ f_0^* \doteq \min \{ f_0(x) : f_i(x) \geq 0, \ 1 \leq i \leq m, \ x \in \mathbb{R}^n \}, \]

where each \( f_i(x) \) is a polynomial i.e. \( f_i(x) = \sum_{\pi \in S(i)} a_{i,\pi} x^{\pi} \).

- Each \( \pi \) is a tuple \( \pi_1, \pi_2, \ldots, \pi_n \) of nonnegative integers, and \( x^{\pi} \doteq x_1^{\pi_1} x_2^{\pi_2} \ldots x_n^{\pi_n} \).
- Each \( S(i) \) is a finite set of tuples, and the \( a_{i,\pi} \) are reals.

Moment Relaxations

- Introduce a variable \( X_\pi \) used to represent each monomial \( x^{\pi} \) of order \( \leq d \), for some integer \( d \).
- This set of monomials includes all of those appearing in the polynomial optimization problem as well as \( x^0 = 1 \).
Higher-order SDP relaxations

Consider the polynomial optimization problem

\[ f_0^* \doteq \min \{ f_0(x) : f_i(x) \geq 0, \ 1 \leq i \leq m, \ x \in \mathbb{R}^n \}, \]

where each \( f_i(x) \) is a polynomial i.e. \( f_i(x) = \sum_{\pi \in S(i)} a_{i,\pi} x^\pi \).

- Each \( \pi \) is a tuple \( \pi_1, \pi_2, \ldots, \pi_n \) of nonnegative integers, and \( x^\pi = x_1^{\pi_1} x_2^{\pi_2} \ldots x_n^{\pi_n} \).
- Each \( S(i) \) is a finite set of tuples, and the \( a_{i,\pi} \) are reals.

Moment Relaxations

- Introduce a variable \( X_\pi \) used to represent each monomial \( x^\pi \) of order \( \leq d \), for some integer \( d \).
- This set of monomials includes all of those appearing in the polynomial optimization problem as well as \( x^0 = 1 \).
- If we replace each \( x^\pi \) in the formulation with the corresponding \( X_\pi \) we obtain a linear relaxation.
Higher-order SDP relaxations

Consider the polynomial optimization problem

$$f_0^* = \min \{ f_0(x) : f_i(x) \geq 0, \ 1 \leq i \leq m, \ x \in \mathbb{R}^n \},$$

where each $f_i(x)$ is a polynomial i.e. $f_i(x) = \sum_{\pi \in S(i)} a_{i,\pi} x^{\pi}$.

- Each $\pi$ is a tuple $\pi_1, \pi_2, \ldots, \pi_n$ of nonnegative integers, and $x^{\pi} = x_1^{\pi_1} x_2^{\pi_2} \ldots x_n^{\pi_n}$
- Each $S(i)$ is a finite set of tuples, and the $a_{i,\pi}$ are reals.

Moment Relaxations

- Introduce a variable $X_\pi$ used to represent each monomial $x^{\pi}$ of order $\leq d$, for some integer $d$.
- This set of monomials includes all of those appearing in the polynomial optimization problem as well as $x^0 = 1$.
- If we replace each $x^{\pi}$ in the formulation with the corresponding $X_\pi$ we obtain a linear relaxation.
- Let $X$ denote the vector of all such monomials. Then $XX^T \succeq 0$ and of rank one. The semidefinite constraint strengthens the formulation.
- Further semidefinite constraints are obtained from the constraints.
I need to solve a large nontrivial SDP

\[(SDP): \quad \min F_0 \bullet X \]
\[
s.t. \quad F_i \bullet X \geq b_i \quad i = 1, \ldots, m
\]
\[
X \succeq 0
\]

... what do I do?
I need to solve a large nontrivial SDP

(SDP): \[ \begin{align*}
\min & \quad F_0 \cdot X \\
\text{s.t.} & \quad F_i \cdot X \geq b_i \quad i = 1, \ldots, m \\
& \quad X \succeq 0
\end{align*} \]

... what do I do? run away even faster

Answer: use **structured sparsity**, if you can
I need to solve a large nontrivial SDP

(SDP): \[
\begin{align*}
\min & \quad F_0 \cdot X \\
\text{s.t.} & \quad F_i \cdot X \geq b_i \quad i = 1, \ldots, m \\
& \quad X \succeq 0
\end{align*}
\]

... what do I do? run away even faster

Answer: use \textit{structured sparsity}, if you can

→ How did power grids develop over time?
Modern grids are very sparse, and “tree-like”
Informal definition

A graph has small *treewidth* if it can be formed by glueing together small blobs (subnetworks) in a tree-like fashion.

- Modern grids have “small” tree-width
- SDP relaxations reflect this fact
Back to ACOPF

\[ V_k = \hat{V}_k e^{j\theta_k^V} = e_k + jf_k, \]

\[ I_{km} = \mathbf{y}_{\{k,m\}} (V_k - V_m), \quad \mathbf{y}_{\{k,m\}} = \text{admittance of } km. \]

\[ p_{km} = \Re (V_k I_{km}^*) \quad q_{km} = \Im (V_{km} I_{km}^*) \]

\[ \hat{V}_k^{\min} \leq |V_k| \leq \hat{V}_k^{\max} \quad \forall \ k \]

Network Inequalities

\[ \hat{P}_k^{\min} \leq \sum_{km \in \delta(k)} p_{km} \leq \hat{P}_k^{\max} \quad \forall \ k \]

\[ \hat{Q}_k^{\min} \leq \sum_{km \in \delta(k)} q_{km} \leq \hat{Q}_k^{\max} \quad \forall \ k \]
Informal definition

A graph has small *treewidth* if it can be formed by glueing together small blobs (subnetworks) in a tree-like fashion.

- Modern grids have “small” tree-width
- SDP relaxations reflect this fact
- SDP algorithms can leverage this fact
Crimes against computers

\[
\begin{align*}
\text{max } & \quad y \\
\text{s.t. } & \quad 1000 \, y + x \leq 1000 \tag{10a} \\
& \quad 10000 \, \delta \geq 1 \tag{10b} \\
& \quad \delta \leq 10 \, a \tag{10c} \\
& \quad a \leq 10 \, b \tag{10d} \\
& \quad b \leq 10 \, c \tag{10e} \\
& \quad c \leq 10 \, d \tag{10f} \\
& \quad d \leq 10 \, x \tag{10g} \\
\end{align*}
\]

\( y \) \text{ binary, all other variables \( \geq 0 \)
Crimes against computers

\[
\begin{align*}
\text{max } & \quad y \\
\text{s.t.} & \quad 1000y + x \leq 1000 & (11a) \\
& \quad 10000 \delta \geq 1 & (11b) \\
& \quad \delta \leq 10a & (11c) \\
& \quad a \leq 10b & (11d) \\
& \quad b \leq 10c & (11e) \\
& \quad c \leq 10d & (11f) \\
& \quad d \leq 10x & (11g) \\
\end{align*}
\]

\(y\) binary, all other variables \(\geq 0\)

Value = 0
More crimes against computers

\[
\text{max} \quad 20x_2 - 20s_5 - 20s_6 + 2s_7 + s_5^2
\]

\[
s.t. \quad (x_1 - 1)^2 + x_2^2 \geq 3 + \frac{\phi}{10} \quad (12a)
\]
\[
(x_1 + 1)^2 + x_2^2 \geq 3 \quad (12b)
\]
\[
\frac{1}{10}x_1^2 + x_2^2 \leq 2 \quad (12c)
\]
\[
10\delta + 10\phi^2 \geq 1 \quad (12d)
\]
\[
-10a + \delta + 10\phi^2 \leq 0
\]
\[
-10b + a + 10\phi^2 \leq 0
\]
\[
-10c + b + 10\phi^2 \leq 0
\]
\[
-10d + c + 10\phi^2 \leq 0
\]
\[
-10e + d + 10\phi^2 + 10s_5^2 = 0 \quad (12e)
\]
\[
-10f + e + 10\phi^2 + 10s_6^2 = 0
\]
\[
-10g + f + 10\phi^2 + 10s_7^2 = 0
\]
\[
-10\phi + g + 10\phi^2 \leq 0 \quad (12f)
\]
What’s going on?

$$\begin{align*}
\text{max} & \quad x_2 \\
\text{s.t.} & \quad (x_1 - 1)^2 + x_2^2 \geq 3 \\
& \quad (x_1 + 1)^2 + x_2^2 \geq 3 \\
& \quad \frac{x_1^2}{10} + x_2^2 \leq 2
\end{align*}$$
What’s going on?

\[
\begin{align*}
\text{max } & \quad x_2 \\
\text{s.t. } & \quad (x_1 - 1)^2 + x_2^2 \geq 3 + \phi \quad (\phi > 0) \\
& \quad (x_1 + 1)^2 + x_2^2 \geq 3 \\
& \quad \frac{x_1^2}{10} + x_2^2 \leq 2
\end{align*}
\]