Solving QCQPs

Daniel Bienstock, Columbia University

Quadratically constrained, quadratic programming:

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

Here,

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

is a general quadratic

Each M_i is $n \times n$, wlog symmetric

Folklore result: QCQP is NP-hard

Folklore result: QCQP is NP-hard

Let $w_1, w_2, \ldots, w_n$ be **integers**, and consider:

$$\begin{aligned} W^* \doteq \min\ & -\sum_i x_i^2 \\ \text{s.t.}\ & \sum_i w_i\, x_i = 0, \\ & -1 \le x_i \le 1, \quad 1 \le i \le n. \end{aligned}$$

Folklore result: QCQP is NP-hard

Let $w_1, w_2, \ldots, w_n$ be **integers**, and consider:

$$
\begin{aligned}
W^* \doteq \min \; & -\sum_i x_i^2 \\
\text{s.t.} \; & \sum_i w_i x_i = 0, \\
& -1 \leq x_i \leq 1, \quad 1 \leq i \leq n.
\end{aligned}
$$

$\boldsymbol{W^* = -n}$, iff there exists a subset $J \subseteq \{1, \ldots, n\}$ with

$$\sum_{j \in J} w_j = \sum_{j \notin J} w_j$$

Take any $\{-1, 1\}$-linear program

Take any $\{-1,1\}$-linear program

$$\begin{aligned} \min\ & c^T x \\ \text{s.t.}\ & Ax = b \\ & x \in \{-1,1\}^n. \end{aligned}$$

Take any $\{-1,1\}$-linear program

$$\begin{aligned} \min\ & c^T x \\ \text{s.t.}\ & Ax = b \\ & x \in \{-1,1\}^n. \end{aligned}$$

$\rightarrow$

$$\begin{aligned} \min\ & c^T x \ - \ \boldsymbol{M} \sum_j x_j^2 \\ \text{s.t.}\ & Ax = b \\ & -1 \le x_j \le 1, \ \ 1 \le j \le n. \end{aligned}$$

(and many other similar transformations)

Observation

Any instance of bounded-variable QCQP

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

with a **fixed** number of distinct bilinear terms can be solved in polynomial time.

Observation

Any instance of bounded-variable QCQP

$$\begin{array}{ll} \min & f_0(x) \\ \text{s.t.} & f_i(x) \;\le\; 0, \quad 1 \le i \le m \\ & x \in \mathbb{R}^n, \quad 0 \le x_j \le 1 \quad \forall j \end{array}$$

with a **fixed** number of distinct bilinear terms can be solved in polynomial time.

$$0 \le x_i \le 1 \;\Rightarrow\; \quad x_i \;\approx\; \textstyle\sum_{k=1}^{N} 2^{-k} y_{ik}, \quad y_i \in \{0,1\}^N \quad \text{(Glover, 1970s)}$$

Observation

Any instance of bounded-variable QCQP

$$\begin{array}{ll} \min & f_0(x) \\ \text{s.t.} & f_i(x) \ \le \ 0, \quad 1 \le i \le m \\ & x \in \mathbb{R}^n, \quad 0 \le x_j \le 1 \quad \forall j \end{array}$$

with a **fixed** number of distinct bilinear terms can be solved in polynomial time.

$0 \le x_i \le 1 \ \Rightarrow \quad x_i \ \approx \ \sum_{k=1}^N 2^{-k} y_{ik}, \quad y_{ik} \in \{0,1\}^N$ (Glover, 1970s)

$$\begin{aligned} x_i x_j \ &= \ \sum_{k=1}^{N} 2^{-k} w_{ijk} \ + \ O(2^{-N}) \\ x_j - 1 + y_{ik} \ &\le \ w_{ijk} \ \le \ x_j \\ 0 \ &\le \ w_{ijk} \ \le \ y_{ik} \end{aligned}$$

Even more general

Solving systems of polynomial equations:

Problem: given polynomials $p_i : \mathbb{R}^n \to \mathbb{R}$, for $1 \leq i \leq m$
find $x \in \mathbb{R}^n$ s.t. $p_i(x) = 0, \forall\, i$

Even more general

Solving systems of polynomial equations:

Problem: given polynomials $p_i : \mathbb{R}^n \to \mathbb{R}$, for $1 \leq i \leq m$
find $x \in \mathbb{R}^n$ s.t. $p_i(x) = 0, \forall\, i$

Example: find a root for $3v^6w - v^4 + 7 = 0$.

Even more general

Solving systems of polynomial equations:

Problem: given polynomials $p_i : \mathbb{R}^n \to \mathbb{R}$, for $1 \le i \le m$
find $x \in \mathbb{R}^n$ s.t. $p_i(x) = 0, \forall\, i$

Example: find a root for $3v^6w - v^4 + 7 = 0$.

Equivalent to the system on variables v, v_2, v_4, v_6, w, y and c:

$$
\begin{aligned}
c^2 &= 1 \\
v^2 - cv_2 &= 0 \\
v_2^2 - cv_4 &= 0 \\
v_2v_4 - cv_6 &= 0 \\
v_6w - cy &= 0 \\
3cy - cv_4 &= -7
\end{aligned}
$$

Smale's 17^{th} problem

Can a zero of n polynomial equations on n unknowns
be found **approximately**,
on the average in polynomial time,
with a **uniform** algorithm?

(but we are cheating)

Smale's 17^{th} problem

Can a zero of n polynomial equations on n unknowns be found **approximately**, **on the average** in polynomial time, with a **uniform** algorithm?

(but we are cheating)

- Approximately?
- On the average?
- Uniform algorithm?

“Approximately”

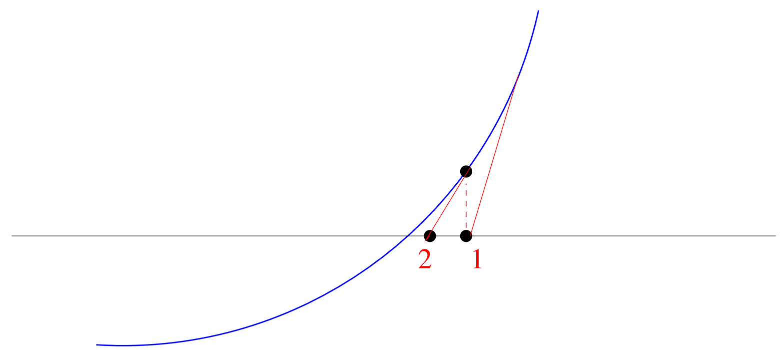
“Approximately”

Q: How do practitioners ~~and other lesser folk~~ solve systems of nonlinear equations?

“Approximately”

Q: How do practitioners ~~and other lesser folk~~ solve systems of nonlinear equations?

A: Newton-Raphson, of course!

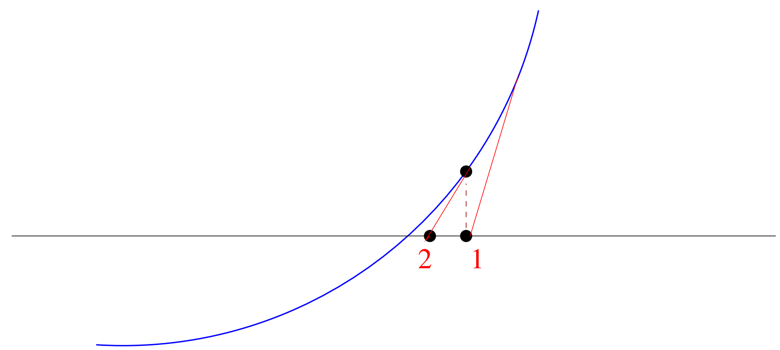


$\rightarrow$ If we start near a solution, quadratic convergence

“Approximately”

Q: How do practitioners ~~and other lesser folk~~ solve systems of nonlinear equations?

A: Newton-Raphson, of course!



$\longrightarrow$ If we start near a solution, quadratic convergence

“Approximate” solution to a system of polynomials:

a point in the region of quadratic convergence (to a solution)

Smale's 17^{th} problem

Can a zero of n polynomial equations on n unknowns be found
approximately,
on the average in polynomial time,
with a **uniform** algorithm?

(but we are cheating)

- Approximately?
- **On the average?**
- Uniform algorithm?

“On the average” in polynomial time

A QCQP could be **quite** difficult!

e.g., a unique feasible solution, which additionally is an irrational vector

“On the average” in polynomial time

A QCQP could be **quite** difficult!
e.g., a unique feasible solution, which additionally is an irrational vector

but a “nearby” problem instance could be much easier

“On the average” in polynomial time

A QCQP could be **quite** difficult!
e.g., a unique feasible solution, which additionally is an irrational vector

but a “nearby” problem instance could be much easier

- View a problem as a vector in an appropriate space

“On the average” in polynomial time

A QCQP could be **quite** difficult!
e.g., a unique feasible solution, which additionally is an irrational vector

but a “nearby” problem instance could be much easier

- View a problem as a vector in an appropriate space
- Endow that space with an appropriate metric
  (Bombieri-Weyl Hermitian product)

“On the average” in polynomial time

A QCQP could be **quite** difficult!
e.g., a unique feasible solution, which additionally is an irrational vector

but a “nearby” problem instance could be much easier

- View a problem as a vector in an appropriate space
- Endow that space with an appropriate metric
  (Bombieri-Weyl Hermitian product)
- In that space, uniformly sample a ball (of appropriate radius) around a given problem

“On the average” in polynomial time

A QCQP could be **quite** difficult!
e.g., a unique feasible solution, which additionally is an irrational vector

but a “nearby” problem instance could be much easier

- View a problem as a vector in an appropriate space
- Endow that space with an appropriate metric (Bombieri-Weyl Hermitian product)
- In that space, consider the set of problems given by a ball (of appropriate radius) around a given problem
- We want the algorithm to run in polynomial time, on average, in that ball

Smale's 17^{th} problem

Can a zero of n polynomial equations on n unknowns be found **approximately**,
on the average in polynomial time,
with a **uniform** algorithm?

(but we are cheating)

- Approximately?
- On the average?
- **Uniform algorithm?** When is an algorithm non-uniform?

Smale's 17^{th} problem

Can a zero of n polynomial equations on n unknowns be found **approximately**, **on the average** in polynomial time, with a **uniform** algorithm?

(but we are cheating)

- Approximately?
- On the average?
- **Uniform algorithm?** When is an algorithm non-uniform?

Blum, Shub, Smale (89), Blum, Cucker, Shub, Smale (98)

First version: A non-uniform algorithm specifies the existence of an algorithm *for each input size.*

As such, we cannot write a "program" that implements the algorithm.

It is more a proof of existence of an algorithm for each input size.

Smale's 17^{th} problem

> Can a zero of n polynomial equations on n unknowns be found
> **approximately**,
> **on the average** in polynomial time,
> with a **uniform** algorithm?

(but we are cheating)

- Approximately?
- On the average?
- **Uniform algorithm?** When is an algorithm non-uniform?

Blum, Shub, Smale (89), Blum, Cucker, Shub, Smale (98)
Bürgisser, Cucker (2012)

Second version: A uniform algorithm

- allows operations over real numbers
- at unit cost per operation
- with infinite precision

Smale's 17^{th} problem

Can a zero of n polynomial equations on n unknowns be found **approximately**, **on the average** in polynomial time, with a **uniform** algorithm?

(but we are cheating)

- Approximately?
- On the average?
- **Uniform algorithm?** When is an algorithm non-uniform?

Blum, Shub, Smale (89), Blum, Cucker, Shub, Smale (98)
Bürgisser, Cucker (2012)

Second version: A uniform algorithm

- allows operations over real numbers
- at unit cost per operation
- with infinite precision

Smale's 17^{th} problem

Can a zero of n polynomial equations on n unknowns be found
approximately,
on the average in polynomial time,
with a **uniform** algorithm?

(but we are cheating)

- Approximately?
- On the average?
- **Uniform algorithm?** When is an algorithm non-uniform?

Blum, Shub, Smale (89), Blum, Cucker, Shub, Smale (98)
Bürgisser, Cucker (2012)

Second version: A uniform algorithm

- allows operations over real numbers
- at unit cost per operation
- with infinite precision
- **Not!** the usual bit-model of computation

Smale's 17^{th} problem

> Can a zero of n polynomial equations on n unknowns be found **approximately**,
> **on the average** in polynomial time,
> with a **uniform** algorithm?

(but we are cheating)

- Beltrán and Pardo (2009) – a randomized (Las Vegas) uniform algorithm that computes an approximate zero in *expected* polynomial time
- Bürgisser, Cucker (2012) – a deterministic $O(n^{\log\log n})$ (uniform) algorithm for computing approximate zeros
- **Techniques:** Homotopy (path-following method solving a sequence of problems), Newton's method

Smale's 17^{th} problem

Can a zero of n polynomial equations on n unknowns be found **approximately**, **on the average** in polynomial time, with a **uniform** algorithm?

(but we are cheating)

- Beltrán and Pardo (2009) – a randomized (Las Vegas) uniform algorithm that computes an approximate zero in *expected* polynomial time
- Bürgisser, Cucker (2012) – a deterministic $O(n^{\log\log n})$ (uniform) algorithm for computing approximate zeros
- **Techniques:** Homotopy (path-following method solving a sequence of problems), Newton's method

But we are cheating: All of this is over $\mathbb{C}^n$, not $\mathbb{R}^n$

Smale's 17^{th} problem

> Can a zero of n polynomial equations on n unknowns be found **approximately**,
> **on the average** in polynomial time,
> with a **uniform** algorithm?

(but we are cheating)

- Beltrán and Pardo (2009) – a randomized (Las Vegas) uniform algorithm that computes an approximate zero in *expected* polynomial time
- Bürgisser, Cucker (2012) – a deterministic $O(n^{\log\log n})$ (uniform) algorithm for computing approximate zeros
- **Techniques:** Homotopy (path-following method solving a sequence of problems), Newton's method

But we are cheating: All of this is over $\mathbb{C}^n$, not $\mathbb{R}^n$

So what can be done over the reals?

Take any $\{-1,1\}$-linear program

$$\begin{aligned} \min \quad & c^T x \\ \text{s.t.} \quad & Ax = b \\ & x \in \{-1,1\}^n. \end{aligned}$$

$\rightarrow$

$$\begin{aligned} \min \quad & c^T x \; - \; \boldsymbol{M} \sum_j x_j^2 \\ \text{s.t.} \quad & Ax = b \\ & -1 \leq x_j \leq 1, \quad 1 \leq j \leq n. \end{aligned}$$

- Fixed number of linear constraints?
- Fixed number of quadratic constraints?
- Non-convex quadratic constraints?

What can be done in polynomial time?

(B. and Alex Michalka, SODA 2014)

$$\begin{aligned} \min \quad & x^T Q x + c^T x \\ \text{s.t.} \quad & \|x - \mu_h\| \le r_h, \quad h \in S, \\ & \|x - \mu_h\| \ge r_h, \quad h \in K, \\ & x \in P \doteq \{\, x \in \mathbb{R}^n \,:\, Ax \le b \,\} \end{aligned}$$

Theorem.
For each fixed $|S|$, $|K|$ can be solved in polynomial time if either

(1) $|S| \ge 1$ and polynomially large number of faces of P intersect

$$\bigcap_{h \in S} \{x \in \mathbb{R}^n \,:\, \|x - \mu_h\| \le r_h\},$$

or

(2) $|S| = 0$ and the number of rows of A is bounded.

(B. and Alex Michalka, SODA 2014)

$$\begin{aligned} \min \quad & x^T Q x + c^T x \\ \text{s.t.} \quad & \|x - \mu_h\| \leq r_h, \quad h \in S, \\ & \|x - \mu_h\| \geq r_h, \quad h \in K, \\ & x \in P \doteq \{\, x \in \mathbb{R}^n \,:\, Ax \leq b \,\} \end{aligned}$$

Theorem.
For each fixed $|S|$, $|K|$ can be solved in polynomial time if either

(1) $|S| \geq 1$ and polynomially large number of faces of P intersecting

$$\bigcap_{h \in S} \{x \in \mathbb{R}^n \,:\, \|x - \mu_h\| \leq r_h\},$$

or

(2) $|S| = 0$ and the number of rows of A is bounded.

Strengthens previous results on the S-Lemma, and trust-region subproblem

The trust-region subproblem:.

$$\begin{aligned} \min \quad & x^T Q x + c^T x \\ \text{s.t.} \quad & \|x - \mu\| \leq r \end{aligned}$$

The trust-region subproblem:.

$$\begin{aligned} \min \quad & x^T Q x + c^T x \\ \text{s.t.} \quad & \|x - \mu\| \leq r \end{aligned}$$

Generalization: CDT (Celis-Dennis-Tapia) problem

$$\begin{aligned} \min \quad & x^T Q_0 x + c_0^T x \\ \text{s.t.} \quad & x^T Q_1 x + c_1^T x + d_1 \ \leq \ 0 \\ & x^T Q_2 x + c_2^T x + d_2 \ \leq \ 0 \end{aligned}$$

where $\boldsymbol{Q_1 \succ 0}$, $\boldsymbol{Q_2 \succ 0}$

Even more general than QCQPs

Barvinok (STOC 1992):

For each fixed $\boldsymbol{p \geq 1}$, there is a polynomial-time algorithm for deciding feasibility of a system

$$\begin{aligned} x^T M_i x &= 0, \quad 1 \leq i \leq p, \\ \|x\| &= 1, \quad x \in \mathbb{R}^n \end{aligned}$$

where the M_i are general matrices.

Even more general than QCQPs

Barvinok (STOC 1992):

For each fixed $\boldsymbol{p \geq 1}$, there is a polynomial-time algorithm for deciding feasibility of a system

$$\begin{aligned} x^T M_i x &= 0, \quad 1 \leq i \leq p, \\ \|x\| &= 1, \quad x \in \mathbb{R}^n \end{aligned}$$

where the M_i are general matrices.

- **Non-constructive.** Algorithm says "yes" or "no."

- **Computational model?** Uniform algorithm? "Real-RAM"?

A (better?) alternative: ϵ-feasibility

For each fixed $\boldsymbol{p \geq 1}$, given a system

$$\begin{aligned} x^T M_i x &= 0, \quad 1 \leq i \leq p, \\ \|x\| &= 1, \quad x \in \mathbb{R}^n \end{aligned}$$

and given $\boldsymbol{0 < \epsilon < 1}$, either

- **Prove** that the system is **infeasible**, or

- **Output** $\hat{\boldsymbol{x}} \in \mathbb{R}^{\boldsymbol{n}}$ with

$$\begin{aligned} -\epsilon \leq x^T M_i &\leq \epsilon, \quad 1 \leq i \leq p, \\ 1 - \epsilon \leq \|\hat{x}\| &\leq 1 + \epsilon, \end{aligned}$$

in time polynomial in the data and in $\log \epsilon^{-1}$.

A (better?) alternative: ϵ-feasibility

For each fixed $\boldsymbol{p \geq 1}$, given a system

$$\begin{aligned} x^T M_i x &= 0, \quad 1 \leq i \leq p, \\ \|x\| &= 1, \quad x \in \mathbb{R}^n \end{aligned}$$

and given $\boldsymbol{0 < \epsilon < 1}$, either

- **Prove** that the system is **infeasible**, or

- **Output** $\boldsymbol{\hat{x} \in \mathbb{R}^n}$ with

$$\begin{aligned} -\epsilon \leq x^T M_i \leq \epsilon, &\quad 1 \leq i \leq p, \\ 1 - \epsilon \leq \|\hat{x}\| \leq 1 + \epsilon, \end{aligned}$$

in time polynomial in the data and in $\log \epsilon^{-1}$.

Two issues: Constructiveness, and ϵ-feasibility

Modification to Barvinok's result

Assume that for each fixed $\boldsymbol{p \geq 1}$, there is an algorithm that given a system

$$\begin{aligned} x^T M_i\, x \;&=\; 0, \quad 1 \leq i \leq p, \\ \|x\| \;&=\; 1, \quad x \in \mathbb{R}^n \end{aligned}$$

and given $\boldsymbol{0 < \epsilon < 1}$, either

- **Proves** that the system is **infeasible**, or

- **Proves** that is $\boldsymbol{\epsilon}$-feasible,

in time polynomial in the data and in $\log \epsilon^{-1}$.

(so still nonconstructive)

Modification to Barvinok's result

Assume that for each fixed $\boldsymbol{p \geq 1}$, there is an algorithm that given a system

$$\begin{aligned} x^T M_i x &= 0, \quad 1 \leq i \leq p, \\ \|x\| &= 1, \quad x \in \mathbb{R}^n \end{aligned}$$

and given $\boldsymbol{0 < \epsilon < 1}$, either

- **Proves** that the system is **infeasible**, or

- **Proves** that is $\boldsymbol{\epsilon}$-feasible,

in time polynomial in the data and in $\log \epsilon^{-1}$.

(so still nonconstructive)

Assuming such an algorithm exists ...

How about systems of quadratic inequalities?

$$f_i(x) \; \leq \; 0 \quad 1 \leq i \leq m \qquad (*)$$

How about systems of quadratic inequalities?

$$f_i(x) \ \leq \ 0 \quad 1 \leq i \leq m \qquad (*)$$

Theorem (cheat)

If at least one f_i is positive-definite, for each fixed m there is a polynomial time to decide feasibility.

How about systems of quadratic inequalities?

$$\boldsymbol{f_i(x) \ \leq \ 0 \quad 1 \leq i \leq m} \qquad (*)$$

Theorem (cheat)
If at least one f_i is positive-definite, for each fixed m there is a polynomial time to decide feasibility.

Assume $f_1(x) \ = \ \|x\|^2 - 1$.

How about systems of quadratic inequalities?

$$f_i(x) \ \leq \ 0 \quad 1 \leq i \leq m \qquad (*)$$

Theorem (cheat)
If at least one f_i is positive-definite, for each fixed m there is a polynomial time to decide feasibility.

Assume $f_1(x) = \|x\|^2 - 1$. And otherwise write $f_i(x) = x^T A_i x + c_i^T x + d_i$.

How about systems of quadratic inequalities?

$$\boldsymbol{f_i(x) \ \le \ 0 \quad 1 \le i \le m} \qquad (*)$$

Theorem (cheat)
If at least one f_i is positive-definite, for each fixed m there is a polynomial time to decide feasibility.

Assume $f_1(x) \ = \ \|x\|^2 - 1$. And otherwise write $f_i(x) = x^T A_i x + c_i^T x + d_i$.

And assume $|f_i(x)| \le U_i$ over $\|x\| \le 1$, $i = 2, \ldots, m$.

How about systems of quadratic inequalities?

$$f_i(x) \ \leq \ 0 \quad 1 \leq i \leq m \qquad (*)$$

Theorem (cheat)
If at least one f_i is positive-definite, for each fixed m there is a polynomial time to decide feasibility.

Assume $f_1(x) \ = \ \|x\|^2 - 1$. And otherwise write $f_i(x) = x^T A_i x + c_i^T x + d_i$.

And assume $|f_i(x)| \leq U_i$ over $\|x\| \leq 1$, $i = 2, \ldots, m$.

Consider the system on variables $\boldsymbol{x} \in \mathbb{R}^n$, $\boldsymbol{v_0} \in \mathbb{R}$, $\boldsymbol{w} \in \mathbb{R}^n$, $\boldsymbol{s} \in \mathbb{R}^n$:

$$\begin{aligned} \|x\|^2 \quad - v_0^2 + s_1^2 \ &= \ 0, && && \text{(1a)} \\ x^T A_i x + c_i^T v_0 x + d_i v_0^2 + s_i^2 \ &= \ 0 && 2 \leq i \leq m, && \text{(1b)} \\ \frac{s_i^2 + w_i^2}{U_i} - v_0^2 \ &= \ 0 && 2 \leq i \leq m, && \text{(1c)} \\ \|x\|^2 + s_1^2 + \sum_{i=2}^{n} \frac{s_i^2 + w_i^2}{U_i} + v_0^2 \ &= \ m + 1. && && \text{(1d)} \end{aligned}$$

Claim: System (1) is feasible iff **(*)** is feasible

Consider a solution to:

$$
\begin{aligned}
\|x\|^2 \quad - v_0^2 + s_1^2 &= 0, && \textbf{(1)}\\
\text{for } 2 \le i \le m: \qquad \frac{s_i^2 + w_i^2}{U_i} - v_0^2 &= 0, && \textbf{(2)}\\
x^T A_i x + c_i^T v_0 x + d_i v_0^2 + s_i^2 &= 0, && \textbf{(3)}\\
\|x\|^2 + s_1^2 + \sum_{i=2}^{n} \frac{s_i^2 + w_i^2}{U_i} + v_0^2 &= m + 1. && \textbf{(4)}
\end{aligned}
$$

Consider a solution to:

$$
\begin{aligned}
& & \|x\|^2 \quad - v_0^2 + s_1^2 &= 0, & \textbf{(1)}\\
& \text{for } 2 \le i \le m: & \frac{s_i^2 + w_i^2}{U_i} - v_0^2 &= 0, & \textbf{(2)}\\
& & x^T A_i x + c_i^T v_0 x + d_i v_0^2 + s_i^2 &= 0, & \textbf{(3)}\\
& & \|x\|^2 + s_1^2 + \sum_{i=2}^{n} \frac{s_i^2 + w_i^2}{U_i} + v_0^2 &= m + 1. & \textbf{(4)}
\end{aligned}
$$

Adding **(1)** and **(2)** yields

$$
\|x\|^2 + s_1^2 + \sum_{i=2}^{n} \frac{s_i^2 + w_i^2}{U_i} = m v_0^2.
$$

Consider a solution to:

$$\begin{aligned} \|x\|^2 \quad - v_0^2 + s_1^2 &= 0, && \textbf{(1)} \\ \text{for } 2 \le i \le m: \qquad \frac{s_i^2 + w_i^2}{U_i} - v_0^2 &= 0, && \textbf{(2)} \\ x^T A_i x + c_i^T v_0 x + d_i v_0^2 + s_i^2 &= 0, && \textbf{(3)} \\ \|x\|^2 + s_1^2 + \sum_{i=2}^{n} \frac{s_i^2 + w_i^2}{U_i} + v_0^2 &= m + 1. && \textbf{(4)} \end{aligned}$$

Adding **(1)** and **(2)** yields

$$\|x\|^2 + s_1^2 + \sum_{i=2}^{n} \frac{s_i^2 + w_i^2}{U_i} \;=\; m v_0^2.$$

and so by **(4)**

$$(m+1)v_0^2 \;=\; (m+1)$$

Consider a solution to:

$$\|x\|^2 \quad - v_0^2 + s_1^2 \;=\; 0, \qquad \textbf{(1)}$$

$$\text{for } 2 \le i \le m: \qquad \frac{s_i^2 + w_i^2}{U_i} - v_0^2 \;=\; 0, \qquad \textbf{(2)}$$

$$x^T A_i x + c_i^T v_0 x + d_i v_0^2 + s_i^2 \;=\; 0, \qquad \textbf{(3)}$$

$$\|x\|^2 + s_1^2 + \sum_{i=2}^{n} \frac{s_i^2 + w_i^2}{U_i} + v_0^2 \;=\; m + 1. \qquad \textbf{(4)}$$

Adding **(1)** and **(2)** yields

$$\|x\|^2 + s_1^2 + \sum_{i=2}^{n} \frac{s_i^2 + w_i^2}{U_i} \;=\; m v_0^2.$$

and so

$$(m+1) v_0^2 \;=\; (m+1)$$

and so by **(4)**

$$v_0^2 \;=\; 1,$$

Consider a solution to:

$$
\begin{aligned}
\|x\|^2 \quad - v_0^2 + s_1^2 &= 0, && \textbf{(1)} \\
\text{for } 2 \le i \le m: \qquad \frac{s_i^2 + w_i^2}{U_i} - v_0^2 &= 0, && \textbf{(2)} \\
x^T A_i x + c_i^T v_0 x + d_i v_0^2 + s_i^2 &= 0, && \textbf{(3)} \\
\|x\|^2 + s_1^2 + \sum_{i=2}^{n} \frac{s_i^2 + w_i^2}{U_i} + v_0^2 &= m + 1. && \textbf{(4)}
\end{aligned}
$$

Adding **(1)** and **(2)** yields

$$
\|x\|^2 + s_1^2 + \sum_{i=2}^{n} \frac{s_i^2 + w_i^2}{U_i} \;=\; m v_0^2.
$$

and so

$$
(m+1) v_0^2 \;=\; (m+1)
$$

and so by **(4)**

$$
v_0^2 \;=\; 1,
$$

and so

$$
\begin{aligned}
\|x\|^2 \quad - 1 &\le 0, \\
x^T A_i x + c_i^T v_0 x + d_i &\le 0 \qquad 2 \le i \le m,
\end{aligned}
$$

which is what we wanted.

Theorem (abridged) System

$$\begin{aligned} \|x\|^2 \quad - 1 &\leq 0, \\ x^T A_i x + c_i^T v_0 x + d_i &\leq 0 \qquad 2 \leq i \leq m, \end{aligned}$$

is ϵ- feasible iff system

$$\begin{aligned} \|x\|^2 \quad - v_0^2 + s_1^2 &= 0, && \textbf{(1)} \\ \text{for } 2 \leq i \leq m: \qquad \frac{s_i^2 + w_i^2}{U_i} - v_0^2 &= 0, && \textbf{(2)} \\ x^T A_i x + c_i^T v_0 x + d_i v_0^2 + s_i^2 &= 0, && \textbf{(3)} \\ \|x\|^2 + s_1^2 + \sum_{i=2}^{n} \frac{s_i^2 + w_i^2}{U_i} + v_0^2 &= m + 1. && \textbf{(4)} \end{aligned}$$

is $O(m\epsilon)$-feasible, and viceversa.

Optimization .

$$
\begin{aligned}
F^* \doteq \min \quad & f_0(x) \\
\text{s.t.} \quad & f_i(x) \ \leq \ 0, \qquad 1 \leq i \leq m.
\end{aligned}
$$

All f_i quadratic, at least one positive definite.

Optimization .

$$\begin{aligned} F^* \doteq \min \quad & f_0(x) \\ \text{s.t.} \quad & f_i(x) \leq 0, \qquad 1 \leq i \leq m \end{aligned}$$

All f_i quadratic, at least one positive definite.

Binary search, $\rightarrow$ a sequence of feasibility problems:

$$\begin{aligned} & f_0(x) \leq B \\ \text{s.t.} \quad & f_i(x) \leq 0, \qquad 1 \leq i \leq m. \end{aligned}$$

Optimization .

$$\begin{aligned} F^* \doteq \min \quad & f_0(x) \\ \text{s.t.} \quad & f_i(x) \leq 0, \qquad 1 \leq i \leq m \end{aligned}$$

All f_i quadratic, at least one positive definite.

Binary search, $\rightarrow$ a sequence of feasibility problems:

$$\begin{aligned} f_0(x) \ & \leq \ B \\ \text{s.t.} \quad f_i(x) \ & \leq \ 0, \qquad 1 \leq i \leq m. \end{aligned}$$

Theorem. In time polynomial in the problem size and $\log \epsilon^{-1}$ we can compute a **value** $\hat{F}$ such that

$$\hat{F} \ \leq \ F^* \ \leq \ \hat{F}.$$

Obtaining explicit solutions

$$\begin{aligned} f_i(x) &\leq 0, \qquad 1 \leq i \leq m, \\ \|x\|^2 &= 1, \end{aligned}$$

(each f_i a quadratic)

Obtaining explicit solutions

$$\begin{aligned} f_i(x) \ &\leq \ 0, \qquad 1 \leq i \leq m, \\ \|x\|^2 \ &= \ 1, \end{aligned}$$

(each f_i a quadratic)

Algorithm:

$\rightarrow$ For $1 \leq j \leq n$ compute*

$$\begin{aligned} \boldsymbol{M_j} \ \doteq \ \max \quad & |\boldsymbol{x_j}| \\ \text{s.t.} \quad & f_i(x) \ \leq \ 0, \qquad 1 \leq i \leq m, \\ & \|x\|^2 \ = \ 1, \end{aligned}$$

in polynomial time (fixed m)

Obtaining explicit solutions

$$\begin{aligned} f_i(x) &\leq 0, \qquad 1 \leq i \leq m, \\ \|x\|^2 &= 1, \end{aligned}$$

(each f_i a quadratic)

Algorithm:

$\longrightarrow$ For $1 \leq j \leq n$ compute*

$$\boldsymbol{M_j} \doteq \max \quad |\boldsymbol{x_j}|$$
$$\text{s.t.} \quad \begin{aligned} f_i(x) &\leq 0, \qquad 1 \leq i \leq m, \\ \|x\|^2 &= 1, \end{aligned}$$

in polynomial time (fixed m)

Assume wlog $\boldsymbol{M_1 = \max\{M_j\}}$.

Obtaining explicit solutions

$$\begin{aligned} f_i(x) &\leq 0, \qquad 1 \leq i \leq m, \\ \|x\|^2 &= 1, \end{aligned}$$

(each f_i a quadratic)

Algorithm:

$\rightarrow$ For $1 \leq j \leq n$ compute*

$$\begin{aligned} \boldsymbol{M_j} \doteq \max \quad & |\boldsymbol{x_j}| \\ \text{s.t.} \quad & f_i(x) \leq 0, \qquad 1 \leq i \leq m, \\ & \|x\|^2 = 1, \end{aligned}$$

in polynomial time (fixed m)

Assume wlog $\boldsymbol{M_1 = \max\{M_j\}}$. M_1 is "big"

Obtaining explicit solutions

$$\begin{aligned} f_i(x) &\leq 0, \qquad 1 \leq i \leq m, \\ \|x\|^2 &= 1, \end{aligned}$$

(each f_i a quadratic)

Algorithm:

$\rightarrow$ For $1 \leq j \leq n$ compute*

$$\begin{aligned} \boldsymbol{M_j} \doteq \max \quad & |\boldsymbol{x_j}| \\ \text{s.t.} \quad & f_i(x) \leq 0, \qquad 1 \leq i \leq m, \\ & \|x\|^2 = 1, \end{aligned}$$

in polynomial time (fixed m)

Assume wlog $\boldsymbol{M_1 = \max\{M_j\}}$. M_1 is "big", i.e. $M_1 \geq n^{-1/2}$.

Obtaining explicit solutions

$$
\begin{aligned}
f_i(x) &\leq 0, \qquad 1 \leq i \leq m, \\
\|x\|^2 &= 1,
\end{aligned}
$$

(each f_i a quadratic)

Algorithm:

$\rightarrow$ For $1 \leq j \leq n$ compute*

$$
\begin{aligned}
\boldsymbol{M_j} \doteq \max \quad & |\boldsymbol{x_j}| \\
\text{s.t.} \quad & f_i(x) \leq 0, \qquad 1 \leq i \leq m, \\
& \|x\|^2 = 1,
\end{aligned}
$$

in polynomial time (fixed m)

Assume wlog $\boldsymbol{M_1 = \max\{M_j\}}$. M_1 is "big", i.e. $M_1 \geq n^{-1/2}$.

$\rightarrow$ **Fix** $\boldsymbol{x_1 = \pm M_1}$

Inductive step

We have fixed

$$
\begin{aligned}
\boldsymbol{x}_1 &= \hat{\boldsymbol{x}}_1 \\
\boldsymbol{x}_2 &= \hat{\boldsymbol{x}}_2 \\
&\cdots \\
\boldsymbol{x}_k &= \hat{\boldsymbol{x}}_k
\end{aligned}
$$

Inductive step

We have fixed

$$\boldsymbol{x_1 = \hat{x}_1, \quad x_2 = \hat{x}_2, \; \ldots \; x_k = \hat{x}_k.}$$

If

$$\sum_{j=1}^{k} \hat{x}_k \;=\; 1,$$

STOP

Inductive step

We have fixed

$$\boldsymbol{x_1 = \hat{x}_1, \quad x_2 = \hat{x}_2, \ \ldots \ \ x_k = \hat{x}_k.}$$

If

$$\sum_{j=1}^{k} \hat{x}_k \ = \ 1,$$

STOP $\longrightarrow$ For $k+1 \leq j \leq n$ compute*

$$\begin{array}{rcll} \boldsymbol{M_j} & \doteq & \max & |\boldsymbol{x_j}| \\ & & \text{s.t.} & f_i(x) \ \leq \ 0, \qquad 1 \leq i \leq m, \\ & & & \|x\|^2 \ = \ 1, \\ & & & x_i \ = \ \hat{x}_i, \qquad 1 \leq i \leq k. \end{array}$$

in polynomial time (fixed m)

Inductive step

We have fixed

$$\boldsymbol{x_1 = \hat{x}_1, \quad x_2 = \hat{x}_2, \; \ldots \quad x_k = \hat{x}_k.}$$

If

$$\sum_{j=1}^{k} \hat{x}_k \; = \; 1,$$

STOP $\rightarrow$ For $k + 1 \leq j \leq n$ compute*

$$\begin{aligned} \boldsymbol{M_j} \;\doteq\; \max \quad & |\boldsymbol{x_j}| \\ \text{s.t.} \quad & f_i(x) \;\leq\; 0, \qquad 1 \leq i \leq m, \\ & \|x\|^2 \;=\; 1, \\ & x_i \;=\; \hat{x}_i, \qquad 1 \leq i \leq k. \end{aligned}$$

in polynomial time (fixed m)

Assume wlog $\boldsymbol{M_{k+1} = \max\{M_j\}}$. $\quad M_{k+1}$ is "big"

$\rightarrow$ **Fix** $\boldsymbol{x_{k+1} \; = \; \pm M_{k+1}}$

Theorem.

For each fixed $m \geq 1$ there is a polynomial-time algorithm that, given an optimization problem

$$\begin{aligned} \min \quad & f_0(x) \doteq x^T Q_0 x + c_0^T x \\ \text{s.t.} \quad & x^T Q_i x + c_i^T x + d_i \leq 0 \quad 1 \leq i \leq m, \end{aligned}$$

where $Q_1 \succ 0$, and $0 < \epsilon < 1$, either

(1) proves that the problem is infeasible,

or

(2) computes an ϵ-feasible vector $\hat{x}$ such that there exists no feasible $x \in \mathbb{R}^n$ with $f_0(x) < f(\hat{x}) - \epsilon$.

The complexity of the algorithm is polynomial in the number of bits in the data and in $\log \epsilon^{-1}$

An application: the Optimal Power Flow problem (OPF)

Input: an undirected graph G.

- For every vertex k, **two** variables: e_k and f_k
- For every edge $\{k, m\}$, **four** (specific) quadratics:

$$
\begin{aligned}
&H^P_{k,m}(e_k, f_k, e_m, f_m), \quad H^Q_{k,m}(e_k, f_k, e_m, f_m)\\
&H^P_{m,k}(e_k, f_k, e_m, f_m), \quad H^Q_{m,k}(e_k, f_k, e_m, f_m).
\end{aligned}
$$

An application: the Optimal Power Flow problem (OPF)

Input: an undirected graph G.

- For every vertex k, **two** variables: e_k and f_k
- For every edge $\{k, m\}$, **four** (specific) quadratics:

$$H^P_{k,m}(e_k, f_k, e_m, f_m), \quad H^Q_{k,m}(e_k, f_k, e_m, f_m)$$
$$H^P_{m,k}(e_k, f_k, e_m, f_m), \quad H^Q_{m,k}(e_k, f_k, e_m, f_m).$$

$$\begin{aligned}
\textbf{min} \quad & \sum_{k \in \mathbb{G}} F_k \left(\sum_{\{k,m\} \in \delta(k)} H^P_{k,m}(e_k, f_k, e_m, f_m) \right) \\
\text{s.t.} \quad & L^P_k \ \le \ \sum_{\{k,m\} \in \delta(k)} H^P_{k,m}(e_k, f_k, e_m, f_m) \ \le \ U^P_k \quad \forall k \\
& L^Q_k \ \le \ \sum_{\{k,m\} \in \delta(k)} H^Q_{k,m}(e_k, f_k, e_m, f_m) \ \le \ U^Q_k \quad \forall k \\
& V^L_k \ \le \ \|(e_k, f_k)\| \ \le \ V^U_k \quad \forall k.
\end{aligned}$$

Function F_k in the objective: convex quadratic

Complexity

Theorem (2011) Lavaei and Low: OPF is (weakly) NP-hard on trees.

Complexity

Theorem (2011) Lavaei and Low: OPF is (weakly) NP-hard on trees.

Theorem (2014) van Hentenryck et al: OPF is (weakly) NP-hard on trees.

Complexity

Theorem (2011) Lavaei and Low: OPF is (weakly) NP-hard on trees.

Theorem (2014) van Hentenryck et al: OPF is (weakly) NP-hard on trees.

Theorem (2007) B. and Verma (2009): OPF is strongly NP-hard on general graphs.

Complexity

Theorem (2011) Lavaei and Low: OPF is (weakly) NP-hard on trees.

Theorem (2014) van Hentenryck et al: OPF is (weakly) NP-hard on trees.

Theorem (2007) B. and Verma (2009): OPF is strongly NP-hard on general graphs.

Recent insight: use the SDP relaxation (Lavaei and Low, 2009 + many others)

$$\begin{aligned}
\min \quad & \sum_{k \in \mathbb{G}} F_k \left(\sum_{\{k,m\} \in \delta(k)} H^P_{k,m}(e_k, f_k, e_m, f_m) \right) \\
\text{s.t.} \quad & L^P_k \ \leq \sum_{\{k,m\} \in \delta(k)} H^P_{k,m}(e_k, f_k, e_m, f_m) \ \leq \ U^P_k \quad \forall k \\
& L^Q_k \ \leq \sum_{\{k,m\} \in \delta(k)} H^Q_{k,m}(e_k, f_k, e_m, f_m) \ \leq \ U^Q_k \quad \forall k \\
& V^L_k \ \leq \ \|(e_k, f_k)\| \ \leq \ V^U_k \quad \forall k.
\end{aligned}$$

Complexity

Theorem (2011) Lavaei and Low: OPF is (weakly) NP-hard on trees.

Theorem (2014) van Hentenryck et al: OPF is (weakly) NP-hard on trees.

Theorem (2007) B. and Verma (2009): OPF is strongly NP-hard on general graphs.

Recent insight: use the SDP relaxation (Lavaei and Low, 2009 + many others)

$$\begin{array}{lcl} \text{e.g.} \quad e_1 f_6 & \rightarrow & w_{e1,f6}, \quad \text{etc} \\ W & \doteq & \{w_{i,j}\}, \quad \text{then} \quad W \succeq 0, \quad W \text{ of rank } 1 \end{array}$$

Complexity

Theorem (2011) Lavaei and Low: OPF is (weakly) NP-hard on trees.

Theorem (2014) van Hentenryck et al: OPF is (weakly) NP-hard on trees.

Theorem (2007) B. and Verma (2009): OPF is strongly NP-hard on general graphs.

Recent insight: use the SDP relaxation (Lavaei and Low, 2009 + many others)

Reformulation of OPF:

$$
\begin{aligned}
\min \quad & F \bullet W \\
\text{s.t.} \quad & A_i \bullet W \ \leq \ b_i \quad i = 1, 2, \ldots \\
& W \succeq 0, \quad W \text{ of rank } 1.
\end{aligned}
$$

Complexity

Theorem (2011) Lavaei and Low: OPF is (weakly) NP-hard on trees.

Theorem (2014) van Hentenryck et al: OPF is (weakly) NP-hard on trees.

Theorem (2007) B. and Verma (2009): OPF is strongly NP-hard on general graphs.

Recent insight: use the SDP relaxation (Lavaei and Low, 2009 + many others)

SDP Relaxation of OPF:

$$\begin{array}{rl} \min & F \bullet W \\ \text{s.t.} & A_i \bullet W \ \leq \ b_i \quad i = 1, 2, \ldots \\ & W \succeq 0. \end{array}$$

Complexity

Theorem (2011) Lavaei and Low: OPF is (weakly) NP-hard on trees.

Theorem (2014) van Hentenryck et al: OPF is (weakly) NP-hard on trees.

Theorem (2007) B. and Verma (2009): OPF is strongly NP-hard on general graphs.

Recent insight: use the SDP relaxation (Lavaei and Low, 2009 + many others)

SDP Relaxation of OPF:

$$\begin{array}{rll} \min & F \bullet W & \\ \text{s.t.} & A_i \bullet W \;\le\; b_i & i = 1, 2, \ldots \\ & W \succeq 0. & \end{array}$$

Fact: The SDP relaxation almost always has a rank-1 solution!!

Complexity

Theorem (2011) Lavaei and Low: OPF is (weakly) NP-hard on trees.

Theorem (2014) van Hentenryck et al: OPF is (weakly) NP-hard on trees.

Theorem (2007) B. and Verma (2009): OPF is strongly NP-hard on general graphs.

Recent insight: use the SDP relaxation (Lavaei and Low, 2009 + many others)

SDP Relaxation of OPF:

$$\begin{aligned} \min \quad & F \bullet W \\ \text{s.t.} \quad & A_i \bullet W \ \leq \ b_i \quad i = 1, 2, \ldots \\ & W \succeq 0. \end{aligned}$$

Fact: The SDP relaxation frequently has a rank-1 solution!!

Complexity

Theorem (2011) Lavaei and Low: OPF is (weakly) NP-hard on trees.

Theorem (2014) van Hentenryck et al: OPF is (weakly) NP-hard on trees.

Theorem (2007) B. and Verma (2009): OPF is strongly NP-hard on general graphs.

Recent insight: use the SDP relaxation (Lavaei and Low, 2009 + many others)

SDP Relaxation of OPF:

$$\begin{aligned} \min \quad & F \bullet W \\ \text{s.t.} \quad & A_i \bullet W \ \leq \ b_i \quad i = 1, 2, \ldots \\ & W \succeq 0. \end{aligned}$$

Fact: The SDP relaxation sometimes has a rank-1 solution!!

Complexity

Theorem (2011) Lavaei and Low: OPF is (weakly) NP-hard on trees.

Theorem (2014) van Hentenryck et al: OPF is (weakly) NP-hard on trees.

Theorem (2007) B. and Verma (2009): OPF is strongly NP-hard on general graphs.

Recent insight: use the SDP relaxation (Lavaei and Low, 2009 + many others)

SDP Relaxation of OPF:

$$\begin{aligned} \min \quad & F \bullet W \\ \text{s.t.} \quad & A_i \bullet W \ \leq \ b_i \quad i = 1, 2, \ldots \\ & W \succeq 0. \end{aligned}$$

Fact: The SDP relaxation sometimes has a rank-1 solution!!
Fact: But it is always very tight!!

Complexity

Theorem (2011) Lavaei and Low: OPF is (weakly) NP-hard on trees.

Theorem (2014) van Hentenryck et al: OPF is (weakly) NP-hard on trees.

Theorem (2007) B. and Verma (2009): OPF is strongly NP-hard on general graphs.

Recent insight: use the SDP relaxation (Lavaei and Low, 2009 + many others)

SDP Relaxation of OPF:

$$\begin{aligned} \min \quad & F \bullet W \\ \text{s.t.} \quad & A_i \bullet W \ \leq \ b_i \quad i = 1, 2, \ldots \\ & W \succeq 0. \end{aligned}$$

Fact: The SDP relaxation sometimes has a rank-1 solution!!
Fact: But it is frequently rather tight!!

Complexity

Theorem (2011) Lavaei and Low: OPF is (weakly) NP-hard on trees.

Theorem (2014) van Hentenryck et al: OPF is (weakly) NP-hard on trees.

Theorem (2007) B. and Verma (2009): OPF is strongly NP-hard on general graphs.

Recent insight: use the SDP relaxation (Lavaei and Low, 2009 + many others)

SDP Relaxation of OPF:

$$\begin{aligned} \min \quad & F \bullet W \\ \text{s.t.} \quad & A_i \bullet W \ \leq \ b_i \quad i = 1, 2, \ldots \\ & W \succeq 0. \end{aligned}$$

Fact: The SDP relaxation sometimes has a rank-1 solution!!
Fact: But it is usually good!!

But: the SDP relaxation is always slow on large graphs

- Real-life grids $\longrightarrow > 10^4$ vertices
- SDP relaxation of OPF does not terminate

But...

But: the SDP relaxation is always slow on large graphs

- Real-life grids $\longrightarrow > 10^4$ vertices
- SDP relaxation of OPF does not terminate

But...

Fact? Real-life grids have **small tree-width**

But: the SDP relaxation is always slow on large graphs

- Real-life grids $\longrightarrow\ > 10^4$ vertices

- SDP relaxation of OPF does not terminate

But...

Fact? Real-life grids have **small tree-width**

Definition 1: A graph has treewidth $\leq \boldsymbol{w}$ if it has a chordal supergraph with clique number $\leq \boldsymbol{w+1}$

But: the SDP relaxation is always slow on large graphs

- Real-life grids $\longrightarrow > 10^4$ vertices
- SDP relaxation of OPF does not terminate

But...

Fact? Real-life grids have **small tree-width**

Definition 2: A graph has treewidth $\leq \boldsymbol{w}$ if it is a subgraph of an intersection graph of subtrees of a tree, with $\leq \boldsymbol{w+1}$ subtrees overlapping at any vertex

But: the SDP relaxation is always slow on large graphs

- Real-life grids $\rightarrow > 10^4$ vertices
- SDP relaxation of OPF does not terminate

But...

Fact? Real-life grids have **small tree-width**

Definition 2: A graph has treewidth $\leq \boldsymbol{w}$ if it is a subgraph of an intersection graph of subtrees of a tree, with $\leq \boldsymbol{w} + \mathbf{1}$ subtrees overlapping at any vertex

(Seymour and Robertson, late 1980s)

But: the SDP relaxation is always slow on large graphs

- Real-life grids $\longrightarrow > 10^4$ vertices

- SDP relaxation of OPF does not terminate

But...

Fact? Real-life grids have **small tree-width**

Cholesky factorization of:

$$A = \begin{pmatrix} * & * & * & & & & & \\ * & * & & * & & & & \\ * & & * & * & * & & * & \\ & * & * & * & & * & & * \\ & & * & & * & * & & \\ & & & * & * & * & & \\ & & * & & & & * & * \\ & & & * & & & * & * \end{pmatrix}$$

But: the SDP relaxation is always slow on large graphs

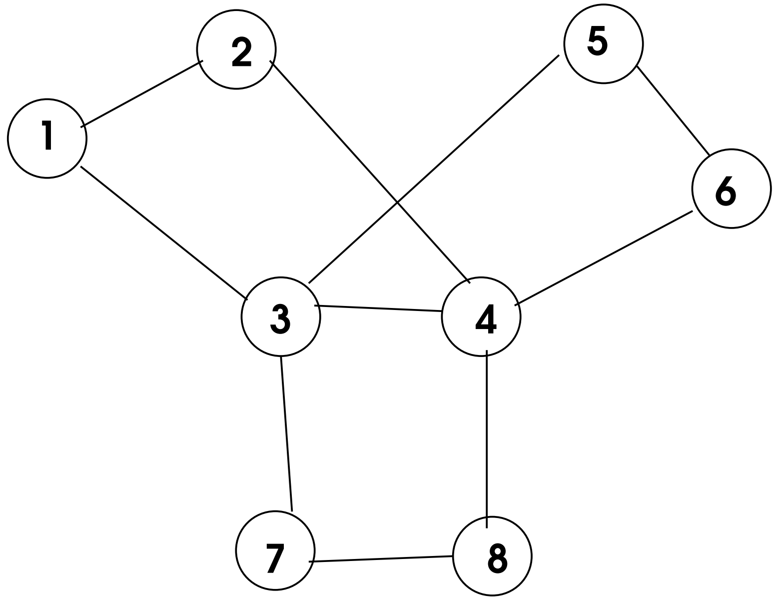
- Real-life grids $\longrightarrow > 10^4$ vertices

- SDP relaxation of OPF does not terminate

But...

Fact? Real-life grids have **small tree-width**

Cholesky factorization of:



But: the SDP relaxation is always slow on large graphs

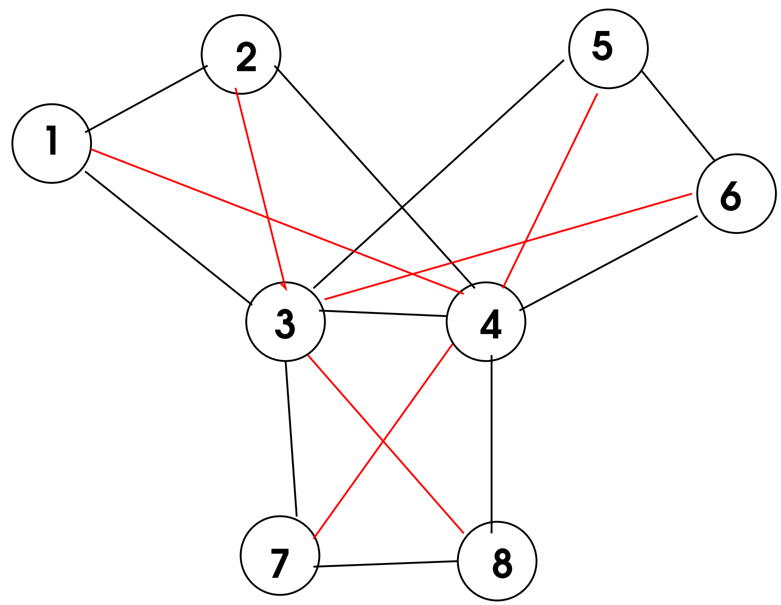
- Real-life grids $\rightarrow > 10^4$ vertices

- SDP relaxation of OPF does not terminate

But...

Fact? Real-life grids have **small tree-width**

Chordal supergraph:



Pivoting order: 1, 2, 5, 6, 7, 8, 3, 4

But: the SDP relaxation is always slow on large graphs

- Real-life grids $\longrightarrow > 10^4$ vertices
- SDP relaxation of OPF does not terminate

But...

Fact? Real-life grids have **small tree-width**

Matrix-completion Theorem gives fast SDP implementations:

Real-life grids with $\approx 3 \times 10^3$ vertices: $\rightarrow$ 20 minutes runtime

OPF

Input: an undirected graph G.

- For every vertex k, **two** variables: e_k and f_k
- For every edge $\{k, m\}$, **four** (specific) quadratics:

$$\begin{array}{ll} H^P_{k,m}(e_k, f_k, e_m, f_m), & H^Q_{k,m}(e_k, f_k, e_m, f_m) \\ H^P_{m,k}(e_k, f_k, e_m, f_m), & H^Q_{m,k}(e_k, f_k, e_m, f_m). \end{array}$$

$$\begin{array}{rl} \mathbf{min} & \displaystyle\sum_{k \in \mathbb{G}} F_k \left(\sum_{\{k,m\} \in \delta(k)} H^P_{k,m}(e_k, f_k, e_m, f_m) \right) \\ \text{s.t.} & L^P_k \ \leq \displaystyle\sum_{\{k,m\} \in \delta(k)} H^P_{k,m}(e_k, f_k, e_m, f_m) \ \leq \ U^P_k \quad \forall k \\ & L^Q_k \ \leq \displaystyle\sum_{\{k,m\} \in \delta(k)} H^Q_{k,m}(e_k, f_k, e_m, f_m) \ \leq \ U^Q_k \quad \forall k \\ & V^L_k \ \leq \ \|(e_k, f_k)\| \ \leq \ V^U_k \quad \forall k. \end{array}$$

Function F_k in the objective: convex quadratic

Graphical QCQP

Input: an undirected graph G.

- For every vertex k, a set of variables: $\{x_j \,:\, j \in I(k)\}$
- For every edge $e = \{k, m\}$, a quadratic

$$H_e(x) \quad = \quad H_e\left(\,\boldsymbol{\{x_j \,:\, j \in I(k) \cup I(m)\}}\right).$$

$$\begin{aligned} \min \quad & \sum_k F_k \left(\sum_{e \in \delta(k)} H_e(x) \right) \\ \text{s.t.} \quad & \sum_{e \in \delta(k)} H_e(x) \;\leq\; b_k \quad \forall k \\ & 0 \leq x_j \leq 1, \quad \forall j \end{aligned}$$

Function F_k in the objective: arbitrary quadratic

Graphical PCPP

Input: an undirected graph G.

- For every vertex k, a set of variables: $\{x_j \,:\, j \in I(k)\}$
- For every edge $e = \{k, m\}$, a **polynomial**

$$P_e(x) \quad = \quad P_e\left(\boldsymbol{\{x_j \,:\, j \in I(k) \cup I(m)}\}\right).$$

$$\begin{aligned} \min \quad & \sum_k F_k \left(\sum_{e \in \delta(k)} P_e(x) \right) \\ \text{s.t.} \quad & \sum_{e \in \delta(k)} P_e(x) \ \leq \ b_k \quad \forall k \\ & 0 \leq x_j \leq 1, \quad \forall j \end{aligned}$$

Function F_k in the objective: arbitrary polynomial

Graphical BPCPP

Input: an undirected graph G.

- For every vertex k, a set of variables: $\{x_j \,:\, j \in I(k)\}$
- For every edge $e = \{k, m\}$, a **polynomial**

$$P_e(x) \quad = \quad P_e\left(\, \boldsymbol{\{x_j \,:\, j \in I(k) \cup I(m)\}}\right).$$

$$\begin{aligned} \min \quad & \sum_k F_k \left(\sum_{e \in \delta(k)} P_e(x) \right) \\ \text{s.t.} \quad & \sum_{e \in \delta(k)} P_e(x) \;\leq\; b_k \quad \forall k \\ & x_j \in \{0, 1\}, \quad \forall j \end{aligned}$$

Function F_k in the objective: arbitrary polynomial

BPCPP

$$\begin{aligned} \min \quad & P_0(x) \\ \text{s.t.} \quad & P_i(x) \ \leq \ b_i \quad i = 1, 2, \ldots, m \\ & x_j \in \{0, 1\}, \quad \forall j \end{aligned}$$

$\rightarrow$ $P_0, P_1, \ldots, P_m$: polynomials

BPCPP

$$\begin{aligned} \min \quad & P_0(x) \\ \text{s.t.} \quad & P_i(x) \;\le\; b_i \quad i = 1, 2, \ldots, m \\ & x_j \in \{0, 1\}, \quad \forall j \end{aligned}$$

$\rightarrow$ the **Clique graph has:**

- A vertex corresponding to each **variable**
- An edge $\{x_i, x_j\}$ if x_i and x_j occur in the same **row**

$$
\begin{aligned}
\textbf{BPCPP:} \min \quad & P_0(x) \\
\text{s.t.} \quad & P_i(x) \ \leq \ b_i \quad i = 1, 2, \ldots, m \\
& x_j \in \{0,1\}, \quad \forall j
\end{aligned}
$$

$\rightarrow$ the **Clique graph has:**

- A vertex corresponding to each **variable**
- An edge $\{x_i, x_j\}$ if x_i and x_j occur in the same **row**

Theorem:

If the clique graph has treewidth $\leq \boldsymbol{w}$, there is an LP with $\boldsymbol{O(2^w m)}$ variables and constraints that solves BPCPP.

$$
\begin{aligned}
\textbf{BPCPP:}\ \min \quad & P_0(x) \\
\text{s.t.} \quad & P_i(x) \ \leq \ b_i \quad i = 1, 2, \ldots, m \\
& x_j \in \{0, 1\}, \quad \forall j
\end{aligned}
$$

$\rightarrow$ the **Clique graph has:**

- A vertex corresponding to each **variable**
- An edge $\{x_i, x_j\}$ if x_i and x_j occur in the same **row**

Theorem:

If the clique graph has treewidth $\leq \boldsymbol{w}$, there is an LP with $\boldsymbol{O(2^w m)}$ variables and constraints that solves BPCPP.

Proof. Lift-and-project techniques.

From GBPCPP to GPCPP

GPCPP:

$$\boldsymbol{F^*} \doteq \min \quad \sum_k F_k \left(\sum_{e \in \delta(k)} P_e(x) \right)$$

$$\text{s.t.} \quad \sum_{e \in \delta(k)} P_e(x) \leq b_k \quad \forall k$$

$$0 \leq x_j \leq 1, \quad \forall j$$

From GBPCPP to GPCPP

$$\textbf{GPCPP:} \qquad \boldsymbol{F^*} \doteq \min \quad \sum_k F_k\left(\sum_{e\in\delta(k)} P_e(x)\right)$$

$$\text{s.t.} \quad \sum_{e\in\delta(k)} P_e(x) \leq b_k \quad \forall k$$

$$0 \leq x_j \leq 1, \quad \forall j$$

$$0 \leq x_i \leq 1 \;\Rightarrow\; x_i = \sum_{k=1}^{N} 2^{-k} y_{ik} + O(2^{-N}), \quad y_i \in \{0,1\}^N$$

From GBPCPP to GPCPP

$$\textbf{GPCPP:} \qquad \boldsymbol{F^*} \doteq \min \quad \sum_k F_k \left(\sum_{e \in \delta(k)} P_e(x) \right)$$

$$\text{s.t.} \quad \sum_{e \in \delta(k)} P_e(x) \leq b_k \quad \forall k$$

$$0 \leq x_j \leq 1, \quad \forall j$$

$$0 \leq x_i \leq 1 \;\Rightarrow\; x_i = \sum_{k=1}^{N} 2^{-k} y_{ik} + O(2^{-N}), \quad y_i \in \{0,1\}^N$$

Theorem:

Given an instance of GPCPP with fixed treewidth of the underlying graph, and $0 < \epsilon < 1$, we can find a vector $\hat{x}$

- in time polynomial in the data and in ϵ^{-1},
- s.t. $\forall k$, $\sum_{e \in \delta(k)} P_e(\hat{x}) \leq b_k + M_k \epsilon$

  (M_k = largest coefficient in $\sum_{e \in \delta(k)} P_e(x)$)
- and $\sum_k F_k \left(\sum_{e \in \delta(k)} P_e(\hat{x}) \right) \leq F^* + M\epsilon$

Mathematical Programming C

The best optimization journal !

Tue.Dec..2.195728.2014@littleboy