Introducing Loraine, software for low-rank semidefinite programming

Michal Kočvara

University of Birmingham

with Soodeh Habibi and Michael Stingl

Programy a algoritmy numerické matematiky 22
Hejnice, June 2024

Poema
Polynomial Optimization, Efficiency through Moments and Algebra

Outline

- Loraine motivation, algorithm and software
- Low-rank SDP problems, numerical results
  - Application 1: Truss topology optimization
  - Application 2: Sensor network localization
  - Application 3: MAXCUT relaxations

Introducing Loraine

Loraine — LOw-RAnk INtErior point method

General purpose SDP solver, particularly efficient for SDP problems with very-low-rank data and/or solutions.

Loraine uses a primal-dual predictor-corrector interior-point method together with (optional) iterative solution of the resulting linear systems.
The iterative solver is a preconditioned Krylov-type method with a preconditioner utilizing low rank of the solution.

Implemented in Matlab and Julia
github.com/kocvara/Loraine.jl
github.com/kocvara/Loraine.m

Linear Semidefinite Optimization, notation

Primal problem

$$\max_{X \in \mathbb{S}^m,\, x_{\text{lin}} \in \mathbb{R}^m} \quad C \bullet X + d^\top x_{\text{lin}}$$

subject to

$$A_i \bullet X + (D^\top x_{\text{lin}})_i = b_i\,, \quad i = 1, \ldots, n$$

$$X \succeq 0, \;\; x_{\text{lin}} \geq 0$$

Dual problem

$$\min_{y \in \mathbb{R}^n,\, S \in \mathbb{S}^m,\, s_{\text{lin}} \in \mathbb{R}^m} \quad b^\top y$$

subject to

$$\sum_{i=1}^{n} y_i A_i + S = C, \;\; S \succeq 0$$

$$Dy + s_{\text{lin}} = d, \;\; s_{\text{lin}} \geq 0$$

Here: $A_i, C \in \mathbb{R}^{m \times m}$, symmetric, $X \bullet Y = \text{trace}(X^\top Y)$

Complexity of IP for Semidefinite Optimization

$$\max_{X \in \mathbb{S}^m} C \bullet X$$

subject to $\quad A_i \bullet X = b_i\,, \quad i = 1, \ldots, n$

$\quad X \succeq 0$

Interior point methods: in every iteration, solve

$$Hy = r, \quad H \in \mathbb{R}^{n \times n}, \text{ full/sparse}$$

	complexity	$n \approx m$	$n \approx m^2$	$n \approx m^4$
H assembly	$nm^3 + n^2m^2$	n^4	n^3	$n^{5/2}$
Cholesky fact.	n^3	n^3	n^3	n^3

	complexity	$n \approx m$	$n \approx m^2$	$m \approx m^4$
H assembly	nm^3	n^4	$n^{5/2}$	$n^{7/4}$
Cholesky fact.	n^α	n^α	n^α	n^α

Loraine ways to efficiency

H assembly:
use low-rank data structure, if available
use a matrix-free (iterative) method (below)

Solution of $Hy = r$:
use (iterative) PCG method with a preconditioner based on low-rank-solution information

Low-rank data

Assume that matrices A_i in $\sum_{i=1}^{n} y_i A_i + S = C, \quad S \succeq 0$ are obtained by

$$A_i = B_i B_i^\top$$

with known data matrices $B_i \in \mathbb{R}^{m \times k}$ and with $k \ll n$.

Using B_i as input matrices, the complexity can be reduced from nm^3 to knm^2 (i.e., nm^2 for rank-one matrices A_i).

If we know that A_i have rank one, the decomposition $A_i = b_i b_i^\top$ is performed by Lorain automatically.

"Simple" to implement but, as to our knowledge, only available in very few SDP codes.

Iterative solvers for $Hy = r$

Use iterative solvers: efficient when $n \gg m$
SDPT3, PENSDP, Bellavia-Gondzio-Porcelli, . . . , this talk

This talk: iterative solver = preconditioned conjugate gradient (CG) method.

What can be gained:

- H assembly: lower complexity (see later), H does not have to be stored in memory
- $Hy = r$ can only be solved approximately, one CG iteration has very low complexity (matrix-vector multiplication)

Drawback:

- H getting (very) ill-conditioned, CG may need very many iterations and may not work at all

Iterative solvers for $Hy = r$

Drawback:

- H getting (very) ill-conditioned, CG may need very many iterations and may not work at all

We need a good preconditioner—problem dependent

Loraine main assumption

Recall:

$$(P) \max_{X,\, x_{\text{lin}}} C \bullet X$$
$$\text{s.t. } A_i \bullet X + (D^\top x_{\text{lin}})_i = b_i \quad \forall i$$
$$X \succeq 0, \quad x_{\text{lin}} \geq 0$$

$$(D) \min_{y,\, S,\, s_{\text{lin}}} c^\top y$$
$$\text{s.t. } \sum_{i=1}^{n} y_i A_i - C = S, \quad S \succeq 0$$
$$Dy + s_{\text{lin}} = d, \quad s_{\text{lin}} \geq 0$$

We assume that the solution X^* has very low rank and develop a preconditioner based on this.

Complementarity: if X^* is low rank then S^* will have almost full rank. Be sure which is your formulation!

Recall: without this assumption, Loraine can still solve the problem but may not be more efficient than other solvers.

Further assumptions

Recall:

$$(P) \max_{X,\, x_{\text{lin}}} C \bullet X$$
$$\text{s.t. } A_i \bullet X + (D^\top x_{\text{lin}})_i = b_i \quad \forall i$$
$$X \succeq 0, \quad x_{\text{lin}} \geq 0$$

$$(D) \min_{y,\, S,\, s_{\text{lin}}} c^\top y$$
$$\text{s.t. } \sum_{i=1}^{n} y_i A_i - C = S, \quad S \succeq 0$$
$$Dy + s_{\text{lin}} = d, \quad s_{\text{lin}} \geq 0$$

Further assumptions:

- Slater condition + strict complementarity
- Sparsity: Define the matrix

$$\mathcal{A} = [\operatorname{svec} A_1, \ldots, \operatorname{svec} A_n].$$

  We assume that matrix-vector products with $\mathcal{A}$ and $\mathcal{A}^T$ may each be applied in $O(n)$ flops and memory.
- "Sparsity" of D: The inverse $(D^\top D)^{-1}$ and matrix-vector product with $(D^\top D)^{-1}$ may each be computed in $\mathcal{O}(n)$ flops and memory.

Low-rank preconditioner for Interior-Point method

In each iteration of the (primal-dual, predictor-corrector) interior-point method we have to solve two systems of linear equations in variable y:

$$(Hy =) \quad \left[\mathcal{A}^\top (W \otimes W)\mathcal{A}\right] y = \text{rhs}$$

Critical observation: If the solution X^* is low-rank, W will be low-rank.

Hence $W = W_0 + UU^\top$ and

$$H = \mathcal{A}^\top (W_0 \otimes W_0)\mathcal{A} + \underbrace{\mathcal{A}^\top (U \otimes Z)}_{V}\underbrace{(U \otimes Z)^\top \mathcal{A}}_{V^\top}$$

where Z is any matrix satisfying $ZZ^\top = 2W_0 + VV^\top$.

Preconditioner: approximate $\mathcal{A}^\top (W_0 \otimes W_0)\mathcal{A}$ by $\tau^2 I$:

$$\mathcal{H}_\alpha = \left(\tau^2 I + D^\top X_{\text{lin}} S_{\text{lin}}^{-1} D\right) + VV^\top .$$

Low-rank preconditioner for Interior-Point method

Preconditioner

$$\mathcal{H}_\alpha = \left(\tau^2 I + D^\top X_{\text{lin}} S_{\text{lin}}^{-1} D\right) + VV^\top .$$

Here $\tau \in \{\text{eigenvalues of } \mathcal{A}^\top (W_0 \otimes W_0)\mathcal{A}\} \to 0$
and V is a narrow matrix with only few columns.

In PCG we ndeed the inverse of $\mathcal{H}_\alpha$!
Using Sherman-Morrison-Woodbury formula:

$$\mathcal{H}_\alpha^{-1} = \tau^{-2}(I - VS^{-1}V^\top(\mathcal{A}^\top\mathcal{A})^{-1})$$

where $S = \tau^2 + V^\top(\mathcal{A}^\top\mathcal{A})^{-1})V$.

Sounds rather complicated but, with some careful coding, it works in practice.

Intermission

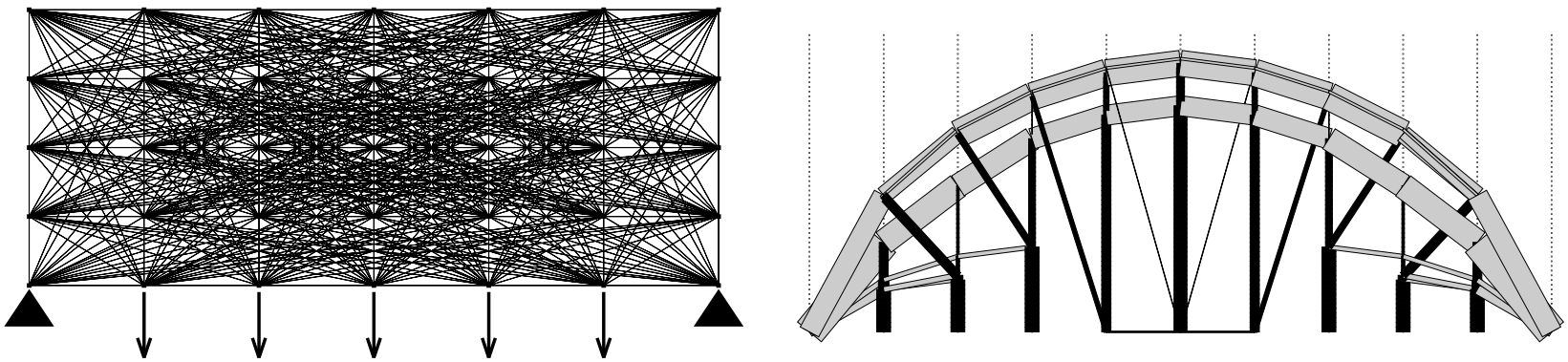
So we have a general purpose SDP solver that is suppsoed to be particularly eficient for problems with low-rank data and low-rank solutions.

Is it?

Application 1: Truss topology optimization

For testing, we use Truss Topology problem which is formulated as large-scale *SDP* with the low-rank solution.

Figure: Truss design: initial ground structure and optimal design.



Application: Truss topology optimization

TTO-SDP problem:

$$\min_{x \in \mathbb{R}^n} \sum_{i=1}^{n} x_i$$

subject to

$$\begin{pmatrix} \gamma & -f^T \\ -f & \sum_{i=1}^{n} K_i x_i \end{pmatrix} \succeq 0$$

$$\underline{x}_i \leq x_i \leq \overline{x}_i, \quad i = 1, \ldots, n$$

- $K_i \in \mathbb{S}^m_+$ are very sparse but $\sum K_i x_i$ is (generally) dense
- $m \ll n$, potentially very large
- if $\underline{x}_i = \varepsilon > 0$ then the dual solution has rank one
- if $\underline{x}_i = 0$ then the dual solution has "almost" rank one: one big outlying eigenvalue, several smaller ones, rest zero

Numerical results

Typical Loraine behaviour (problem tru11):

```
 Number of variables:  7260
 Matrix size(s)      :   221
 Linear constraints : 14520
*** Loraine starts
 it        obj          error        cg_it     time
  1   7.30406207e+02   1.16e+06      4   5     0.05
  2   1.78748113e+04   9.82e+05      1   1     0.04
  3   1.96680257e+04   9.83e+04      1   1     0.04
 ...
 21   9.13602737e-02   1.37e-01      7   7     0.06
 22   8.63200155e-02   1.22e-01      6   7     0.06
 ...
 34   5.96932417e-02   1.67e-04      6   6     0.05
 35   5.96505741e-02   1.74e-05      7   6     0.05
 36   5.96461606e-02   2.17e-06      6   6     0.05
*** Total CG iterations:      344
*** Total CPU time:          1.82 seconds
```

Numerical Results

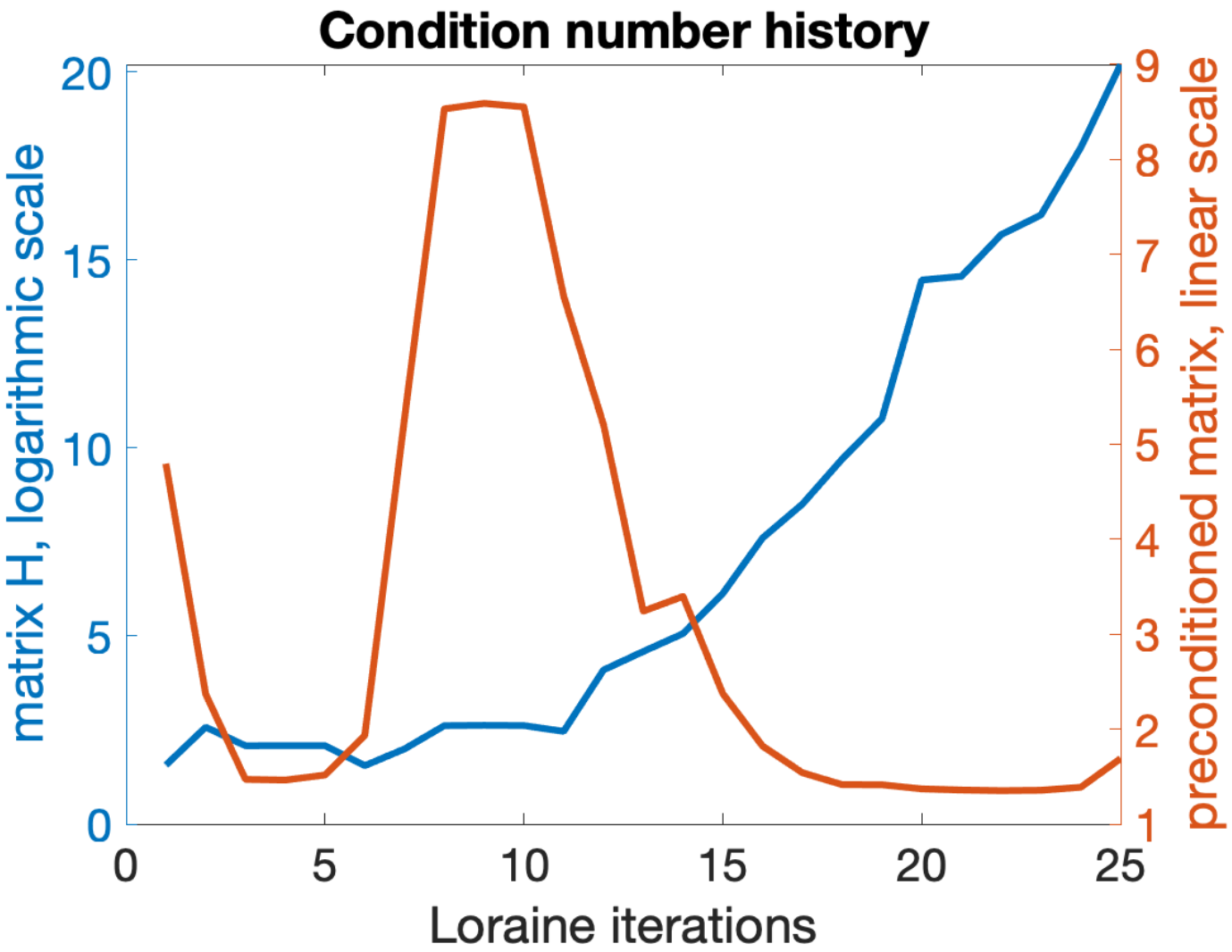
Truss problems:

Mosek (direct solver) vs Loraine (iterative solver)

			Mosek		Loraine	
	variables	LMI+lin	IP iter	time	IP iter	time
tru3	36	13+72	14	0.22	16	0.01
tru5	300	41+600	15	0.23	21	0.08
tru7	1176	85+2352	17	0.78	27	0.24
tru9	3240	145+6480	20	5.6	31	0.65
tru11	7260	221+14520	26	44	36	1.6
tru13	14196	313+28392	29	235	45	4.5
tru15	25200	421+50400	32	1079	52	11
tru17	41616	545+83232	37	4671	53	24
tru19	64980	685+129960	mem	-	64	51
tru21	97020	841+194040	mem	-	65	84
tru23	139656	1013+279312	mem	-	72	189
tru25	195000	1201+390000	mem	-	87	317

iMac with 3.6 GHz 8-Core Intel Core i9 and 40 GB RAM

Numerical results, problem tru7



Numerical results

Truss problems:

Mosek (direct solver) vs Loraine (iterative solver)

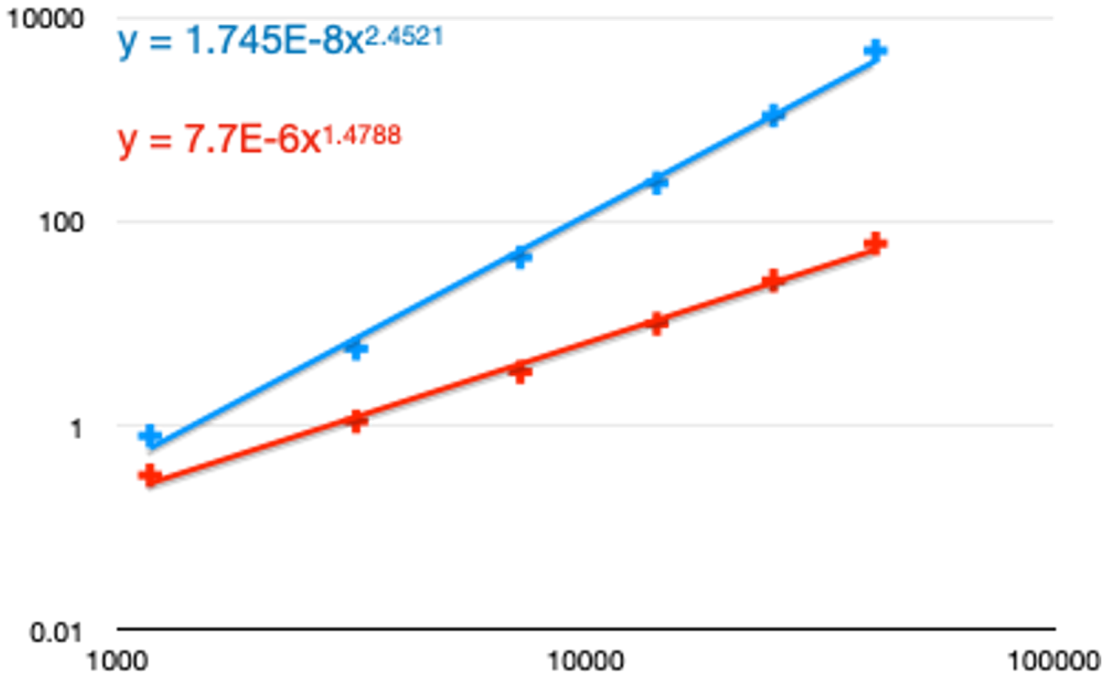


Figure: Mosek vs Loraine elapsed time

Low-rank data: Numerical results

Truss problems: data matrices A_i have rank one!

We compare Mosek (direct solver) and Loraine (direct solver)

	Loraine direct			MOSEK		
problem	iter	time	time/iter	iter	time	time/iter
tru3	16	0.01	0.001	14	0.22	0.02
tru5	21	0.09	0.004	15	0.23	0.02
tru7	27	0.46	0.02	17	0.78	0.05
tru9	31	4.3	0.14	20	5.6	0.28
tru11	36	31	0.86	26	44	1.69
tru13	45	239	5.31	29	235	8.10
tru15	52	1216	23.38	32	1079	33.72
tru17	51	4583	89.86	37	4671	126.24

Numerical results, other software

problem	Loraine direct iter	Loraine direct time	MOSEK iter	MOSEK time	SDPNAL+ iter	SDPNAL+ time	SDPLR iter	SDPLR time
tru3	16	0.01	14	0.22	440	0.5	15	0.01
tru5	21	0.09	15	0.23	9674	29	18	1.7
tru7	27	0.46	17	0.78	6324	55	19	21
tru9	31	4.3	20	5.6	47579	468	20	179
tru11	36	31	26	44	maxit		22	2469
tru13	45	239	29	235			maxit	
tru15	52	1216	32	1079				
tru17	51	4583	37	4671				

Application 2: Sensor network localization (SNL)

(Euclidean distance matrix completion, Graph realization)

We have (in $\mathbb{R}^2$ (or $\mathbb{R}^d$))

n_a points a_i, anchors with known location

n_s points x_i, sensors with unknown location

d_{ij} known Euclidean distance of "close" points within radio range ρ

$$\underline{d}_{ij} = \|x_i - x_j\|, \ (i,j) \in \mathcal{I}_x$$

$$\overline{d}_{kj} = \|a_k - x_j\|, \ (k,j) \in \mathcal{I}_a$$

Goal: Find the positions of the sensors!

In many applications the data d_{ij} is noisy:

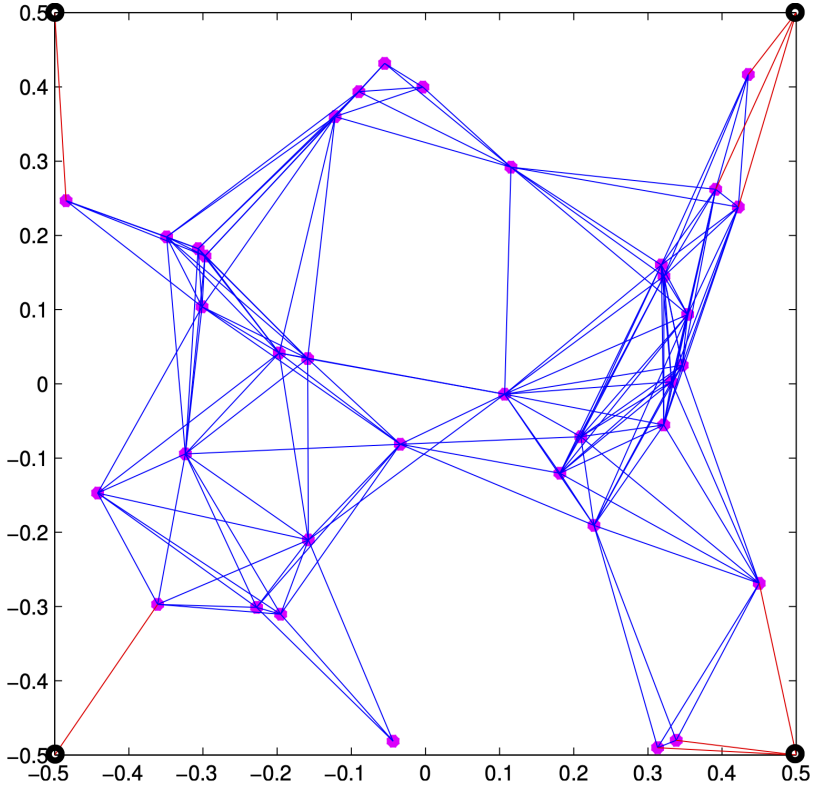
$$d_{ij} = \hat{d}_{ij}|1 + \varepsilon_{ij}|, \quad (i,j) \in \mathcal{I}_x$$

$$\overline{d}_{ik} = \hat{\overline{d}}_{ik}|1 + \varepsilon_{ik}|, \quad (i,j) \in \mathcal{I}_a$$

where $\hat{d}_{ij}$ are the true distances.

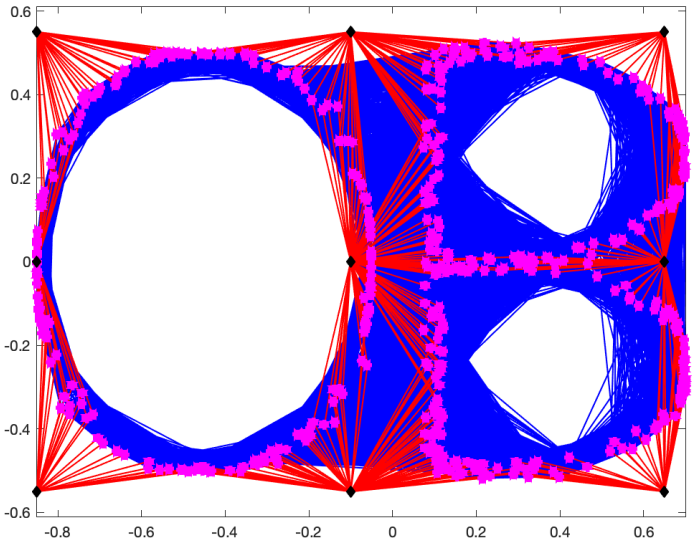
Sensor network localization

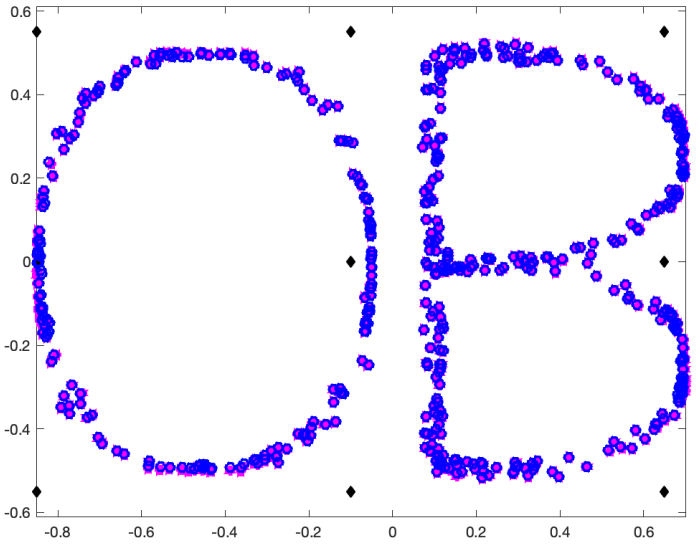
Example, 4 anchors, 36 sensors



Sensor network localization

Example, 9 anchors, 512 sensors





Sensor network localization

Goal: Find the positions of the sensors!

Find $x \in \mathbb{R}^{2\times n_s}$ such that

$$\|x_i - x_j\|^2 = d_{ij}^2, \qquad (i,j) \in \mathcal{I}_x$$
$$\|a_k - x_j\|^2 = \overline{d}_{kj}^2, \qquad (k,j) \in \mathcal{I}_a$$

Infeasible for noisy data. Replaced by least-square problem

$$\min_{x_1,\dots,x_m} \sum_{(i,j)\in\mathcal{I}_x} \left| \|x_i - x_j\|^2 - d_{ij}^2 \right| + \sum_{(i,j)\in\mathcal{I}_a} \left| \|a_k - x_j\|^2 - \overline{d}_{kj}^2 \right|$$

We need global minimum $\rightarrow$ an NP-hard problem.

"Full" SDP relaxation (P. Biswas and Y. Ye, '04)

Let $X = [x_1 \; x_2 \; \dots \; x_{n_s}]$ be a $2 \times n_s$ unknown matrix and

$$Z = \begin{pmatrix} I_d & X \\ X^T & X^T X \end{pmatrix} \succeq 0.$$

Relaxed problem:

$$\min_{Z,u,v} \sum_{(i,j)\in\mathcal{I}_x} (u_{ij} + v_{ij}) + \sum_{(k,j)\in\mathcal{I}_a} (u_{kj} + v_{kj})$$

subject to

$$Z_{1:d,1:d} = I_d$$

$$(0; e_i - e_j)^T Z (0; e_i - e_j) - u_{ij} + v_{ij} = d_{ij}^2, \quad (i,j) \in \mathcal{I}_x$$

$$(a_k; -e_j)^T Z (a_k; -e_j) - u_{kj} + v_{kj} = \overline{d}_{kj}^2, \quad (k,j) \in \mathcal{I}_a$$

$$Z \succcurlyeq 0$$

$$u_{ij}, v_{ij} \geq 0, \quad (i,j) \in \mathcal{I}_x, \qquad u_{kj}, v_{kj} \geq 0, \quad (k,j) \in \mathcal{I}_a$$

If Z low-rank, the SDP problem is in the "right" form for the low-rank preconditioner.

SDP relaxation

So and Ye (2007): for certain "good" problems with no data noise this relaxation is exact, i.e., rank $Z^* = 2$.

For noisy problems, Z^* will have two large eigenvalues and several, perhaps many, small nonzero ones. The problem thus satisfies our low-rank assumption and can be efficiently solved by Loraine.

SNL—Numerical Results

SNL problems of growing dimension with nine anchors and n sensors randomly distributed around the letters O and B. The radio range ρ was set to $\frac{1}{5}$ of the maximal vertical distance of the anchors. The known distances between the sensors were perturbed by 1% Gaussian noise.

				Loraine			MOSEK	
problem	n	m	lin.con.	iter	CG it.	time	iter	time
SNL-16	1677	130	3348	21	395	1.4	13	1.0
SNL-32	6818	258	13630	25	532	7.4	15	21
SNL-48	14886	386	29766	27	676	21	17	150
SNL-64	26767	514	53528	27	705	40	16	611
SNL-80	40840	642	81674	28	1064	84	21	2052
SNL-96	58956	770	117906	29	969	120	27	6579
SNL-112	82137	898	164268	30	1293	224	memory	
SNL-128	109464	1026	218922	30	1316	299		

Application 3: Relaxations of MAXCUT problem

Unconstrained BQP:

Find a global minimum of the non-convex binary problem

$$\min_{x \in \mathbb{R}^s} x^\top Q x \quad \text{subject to} \quad x_i \in \mathcal{B}\,, \quad i = 1, \ldots, s \qquad \text{(BQP)}$$

$Q \in \mathbb{R}^{s \times s}$ symmetric, $\mathcal{B}$ either $\{0, 1\}$ or $\{-1, 1\}$.

We do not assume any sparsity in Q, it is a generally dense matrix.

Technique: Hierarchy of convex conic relaxations

Unconstrained BQP

To find a global optimum, we use Lasserre hierarchy of semidefinite optimization (SDP) problems—relaxations—of growing dimension.
The SDP relaxations have the form

$$\min_{y \in \mathbb{R}^n} q^\top y \qquad \text{(BQP-rel)}$$

$$\text{subject to } M(y) := \sum_{i=1}^{n} y_i M_i - M_0 \succeq 0\,.$$

Here M is a *moment matrix*, a (generally) dense matrix of a very specific form. (For $\omega = 1$, we have $q = \mathrm{svec}(Q)$.)

In particular, if the solution of (BQP) is unique and the order of the relaxation is high enough, then $\mathrm{rank}(M) = 2$.

Dimensions of the relaxations

Theoretical bound on ω to get exact solution is $\lceil s/2 \rceil$
(lower bound: Laurent 2003; upper bound: Fawzi et al. 2016).

This is confirmed by (specially constructed) examples.

This gives

s	ω	matrix size
21	11	784 625
31	16	759 852 346
41	21	$7.5 \cdot 10^{11}$
51	26	$7.5 \cdot 10^{14}$

So problems with $s > 20$ seem unsolvable by this approach.

BUT: in many (most?) numerical examples, $\omega = 2$ delivers global solution!

So, in the following, we will focus on the case $\omega = 2$.

Loraine and relaxed BQP?

Recall:

$$(P) \max_{X,\, x_{\text{lin}}} C \bullet X$$
$$\text{s.t. } A_i \bullet X + (D^\top x_{\text{lin}})_i = b_i \quad \forall i$$
$$X \succeq 0, \quad x_{\text{lin}} \geq 0$$

$$(D) \min_{y,\, S,\, s_{\text{lin}}} c^\top y$$
$$\text{s.t. } \sum_{i=1}^{n} y_i A_i - C = S, \quad S \succeq 0$$
$$Dy + s_{\text{lin}} = d, \quad s_{\text{lin}} \geq 0$$

Loraine assumes that the solution X^* has very low rank.

The SDP relaxation of BQP has the form of (D) with low-rank solution S, just the opposite of our assumption!

Using additional variables and equality constraints, we will reformulate it as (P) with low-rank solution X.

Re-writing the BQP relaxation

The dual problem to

$$\min_{y \in \mathbb{R}^n} q^\top y \qquad \text{(BQP-rel)}$$

$$\text{subject to } M(y) := \sum_{i=1}^{n} y_i M_i - M_0 \succeq 0\,.$$

can be written as

$$\max_{z \in \mathbb{R}^{\tilde{n}}} (\mathrm{svec}(I))^\top z \qquad \text{(BQP-rel-dual)}$$

$$\text{subject to } \mathrm{smat}(z) \succeq 0$$

$$\mathbf{M} z = \tilde{q}\,,$$

where $\widetilde{n} = m(m+1)/2$, $\mathbf{M} = (\mathrm{svec}(M_1), \ldots, \mathrm{svec}(M_n))^T \in \mathbb{R}^{n \times \widetilde{n}}$.

Now the dual solution to (BQP-rel-dual) has rank two, just as Loraine needs.

Handling linear equalities

Problem

$$\max_{z \in \mathbb{R}^{\tilde{n}}} (\mathrm{svec}(I))^\top z \qquad \text{(BQP-rel-dual)}$$
$$\text{subject to } \mathrm{smat}(z) \succeq 0$$
$$\mathbf{M}z = \tilde{q}$$

is now in the right form.

But what about the (many) linear equality constraints?

Interior-point methods do not like them.

Treat them by ℓ_1 penalty:

$$\max_{z \in \mathbb{R}^{\tilde{n}}} (\mathrm{svec}(I))^\top z + \mu \|\mathbf{M}z - \tilde{q}\|_1$$
$$\text{subject to } \mathrm{smat}(z) \succeq 0\,,$$

with an (exact!) penalty parameter $\mu > 0$.

Handling linear equalities

Introduce two new variables, $r \in \mathbb{R}^n$, $s \in \mathbb{R}^n$, satisfying

$$\mathbf{M}z - \tilde{q} = r - s, \quad r \geq 0, \; s \geq 0\,.$$

Using the identity $r = \mathbf{M}z - \tilde{q} + s$ to eliminate variable r, we arrive at our final problem

$$\max_{z \in \mathbb{R}^{\tilde{n}},\, s \in \mathbb{R}^n} (\mathrm{svec}(I))^\top z + \mu \sum_{i=1}^{n} ((\mathbf{M}z - \tilde{q})_i + 2s_i) \quad \text{(BQP-rel-final)}$$

$$\begin{aligned} \text{subject to } & \mathrm{smat}(z) \succeq 0 \\ & \mathbf{M}z - \tilde{q} + s \geq 0 \\ & s \geq 0 \end{aligned}$$

Problem (BQP-rel-final) is now in the required form.

Sparsity assumption

Recall the sparsity assumptions:

- Sparsity: Define the matrix

$$\mathcal{A} = [\operatorname{svec} A_1, \ldots, \operatorname{svec} A_n].$$

  We assume that matrix-vector products with $\mathcal{A}$ and $\mathcal{A}^T$ may each be applied in $O(n)$ flops and memory.

Every matrix A_i contains at most two nonzero elements, hence this is trivially satisfied.

- "Sparsity" of D: The inverse $(D^\top D)^{-1}$ and matrix-vector product with $(D^\top D)^{-1}$ may each be computed in $\mathcal{O}(n)$ flops and memory.

Lemma: There exists a permutation matrix $P \in \mathbb{R}^{\widetilde{n} \times \widetilde{n}}$ such that $P\mathbf{M}^T\mathbf{M}P^T$ is a block diagonal matrix with small full blocks. In particular, $\mathbf{M}^T\mathbf{M}$ is a sparse chordal matrix.

Numerical experiments

Order-two relaxation of the MAXCUT problems

$$\max_x \left\{ \frac{1}{4} \sum_{i,j=1}^{n} a_{ij}(1 - x_i x_j) \mid x_i^2 = 1,\ i = 1, \ldots, n \right\}$$

The solution MAXCUT problem is of rank two only if the problem has a unique global solution.

To avoid the non-uniqueness, we generated undirected, weighted, generally complete graphs with weights randomly distributed between 0 and 12 with 20–50 nodes. It turned out that for all these problems the relaxation order two in the Lasserre hierarchy is already high enough to deliver the optimal solution of the MAXCUT problem.

Numerical experiments, problem sizes

Problems `MAXCUT-<n>`, relaxation order $\omega = 2$; number of variables n, size of the LMI constraint m and number of linear constraints.

BQP size	problem (BQP-rel) variables	problem (BQP-rel) matrix size	problem (BQP-rel-final) variables	problem (BQP-rel-final) matrix size	problem (BQP-rel-final) lin. con.
10	385	56	1 981	56	770
15	1 940	121	9 321	121	3 880
20	6 195	211	28 561	211	12 390
25	15 275	326	68 576	326	30 550
30	31 930	466	140 741	466	63 860
35	59 535	631	258 904	631	119 070
40	102 090	821	439 521	821	204 180
45	164 220	1 036	701 386	1 036	328 440
50	251 175	1 276	1 065 901	1 276	502 350

Numerical experiments, results

Loraine, MOSEK and ADMM in `MAXCUT-<n>` problems

	Loraine (BQP-rel-final)			MOSEK (BQP-rel)		ADMM (BQP-rel-final)	
problem	iter	CG iter	time	iter	time	iter	time
MAXCUT-20	18	559	3.8	6	9	3042	13
MAXCUT-25	20	728	11	7	78	2735	28
MAXCUT-30	21	1032	28	9	607	3537	80
MAXCUT-35	23	2183	96	9	2911	3030	126
MAXCUT-40	27	2275	186	memory		1280	92
MAXCUT-45	25	2521	335			2639	358
MAXCUT-50	24	2540	528			3296	745

iMac with 3.6 GHz 8-Core Intel Core i9 and 40 GB RAM

Numerical experiments, results

Loraine warm-started by ADMM:

After a “few” ADMM iterations, the approximate solution X is still very inaccurate but has the correct rank!

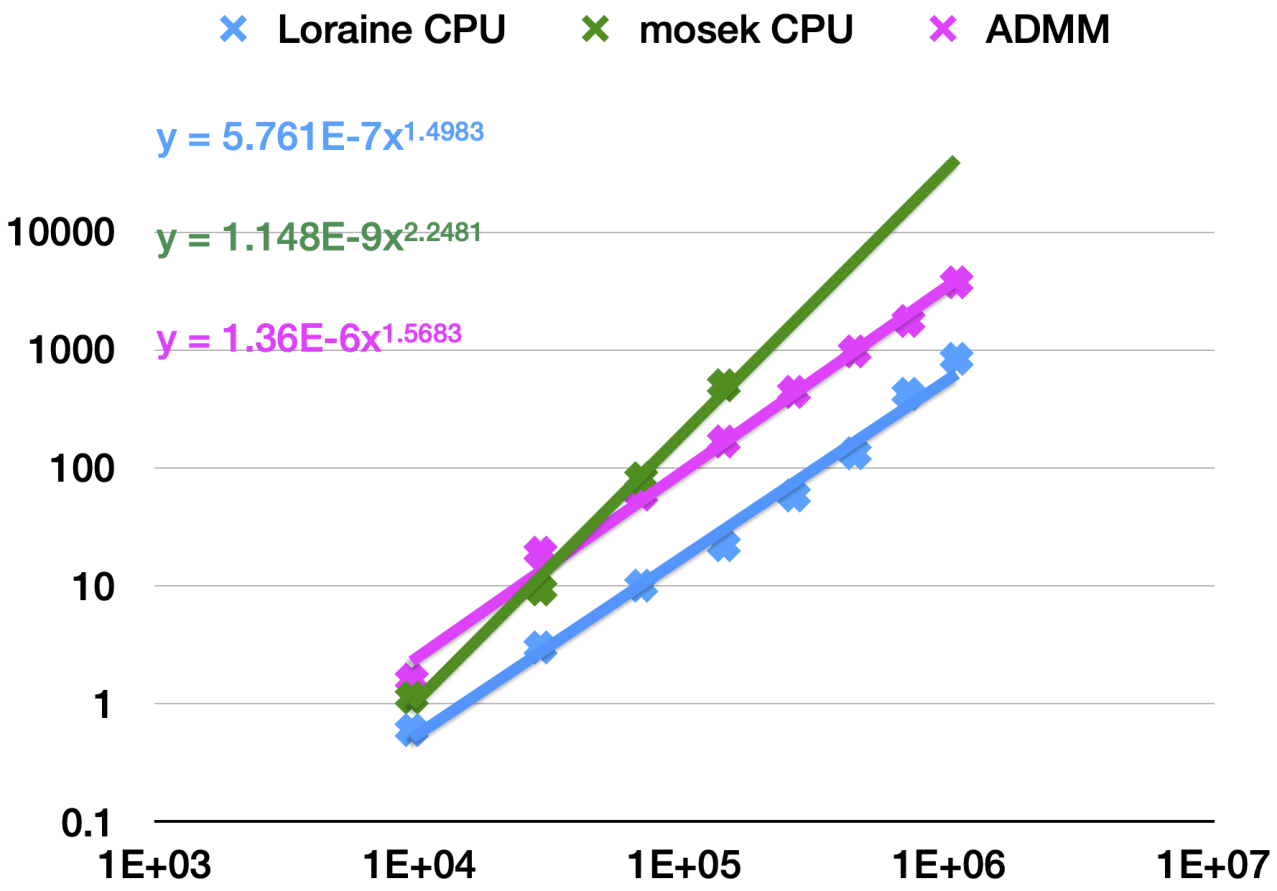
Loraine preconditioner efficient from the (warm) start.

MAXCUT	Loraine		ADMM		ADMM-Loraine			
problem	iter	time	iter	time	iter	timeA	timeL	time
20	18	3.8	3042	13	628+4	3.9	0.8	4.7
25	20	11	2735	28	181+5	2.5	2.9	5.4
30	21	28	3537	80	795+5	20	5.8	26
35	23	96	3030	126	863+4	42	11	53
40	27	186	1280	92	914+7	73	132	205
45	25	335	2639	358	755+4	104	33	137
50	24	528	3296	745	727+5	162	58	220

Numerical results

MAXCUT problems:

Mosek (direct solver) vs Loraine (iterative solver) vs ADMM



Dimension vs elapsed time, log-log scale

∼ THE END ∼

- S. Habibi, M. Kočvara and M. Stingl:
  Loraine – An interior-point solver for low-rank semidefinite programming, https://hal.science/hal-04076509, 2023
- S. Habibi, M. Kočvara and M. Stingl:
  Solving Lasserre relaxations of unconstrained binary quadratic optimization problems by an interior-point method, https://hal.science/hal-04076510, 2023
- github.com/kocvara/Loraine.m
- github.com/kocvara/Loraine.jl (Julia version, integrated into JuMP environment)