TSSOS: a Julia Library to Exploit Sparsity for Large-Scale Polynomial Optimization

Jie Wang (LAAS-CNRS)
Joint work with Victor Magron

7-11 June, MEGA 2021

Content

1. Polynomial optimization and the moment-SOS hierarchy
2. Exploiting sparsity in the moment-SOS hierarchy
3. The usage of `TSSOS`
4. Numerical experiments

Polynomial optimization problem

TSSOS aims to solve the polynomial optimization problem (POP):

$$
(\mathrm{Q}): \quad \begin{array}{rll} f^* := & \inf & f \\ & \text{s.t.} & g_j \geq 0, \quad j = 1, \ldots, m, \\ & & g_j = 0, \quad j = m+1, \ldots, m+l, \end{array}
$$

where $f, g_j \in \mathbb{R}[\mathrm{x}] := \mathbb{R}[x_1, \ldots, x_n]$.

In general, the problem (Q) is non-convex, NP-hard.

What do we care about in polynomial optimization?

- Compute the global optimal value
- Certify global optimality
- Extract global optimal solutions
- Approximate/bound the global optimal value if the exact computation is impossible/unnecessary/expensive

What do we care about in polynomial optimization?

- Compute the global optimal value
- Certify global optimality
- Extract global optimal solutions
- Approximate/bound the global optimal value if the exact computation is impossible/unnecessary/expensive

The moment-SOS hierarchy established by Lasserre has become a powerful tool to handle POPs and possesses nice theoretical properties.

The moment relaxation

The moment relaxation (with relaxation order d) for (Q):

$$
(\mathrm{Q}_d): \quad
\begin{array}{rll}
\theta_d := & \inf & L_{\mathrm{y}}(f) \\
& \mathrm{s.t.} & M_d(\mathrm{y}) \succeq 0, \\
& & M_{d-d_j}(g_j\mathrm{y}) \succeq 0, \quad j = 1, \ldots, m, \\
& & M_{d-d_j}(g_j\mathrm{y}) = 0, \quad j = m+1, \ldots, m+l, \\
& & y_0 = 1,
\end{array}
$$

where $d_j := \lceil \deg(g_j)/2 \rceil$.

This is a semidefinite programming (SDP) problem, solved by interior-point solvers (e.g. `Mosek`) or first-order solvers (e.g. `COSMO`).

The SOS relaxation

For each d, the dual SDP of the moment relaxation is the following SOS relaxation:

$$(\mathrm{Q}_d)^*: \quad \begin{array}{rll} \theta_d^* := & \sup & \lambda \\ & \text{s.t.} & f - \lambda = \sigma_0 + \sum_{j=1}^{m} \sigma_j g_j + \sum_{j=m+1}^{m+l} \tau_j g_j, \\ & & \sigma_j \in \Sigma_{2(d-d_j)}[\mathbf{x}], \quad j = 0, 1, \ldots, m, \\ & & \tau_j \in \mathbb{R}_{2(d-d_j)}[\mathbf{x}], \quad j = m+1, \ldots, m+l. \end{array}$$

$$
\begin{array}{ccccc}
 & & f^* & & \\
 & \geq & & \geq & \\
 & \vdots & & \vdots & \\
 & \geq & & \geq & \\
\text{(the moment relaxation)} & \theta_d & \text{“}=\text{”} & \theta_d^* & \text{(the SOS relaxation)} \\
 & \geq & & \geq & \\
 & \vdots & & \vdots & \\
 & \geq & & \geq & \\
 & \theta_{\underline{d}} & \text{“}=\text{”} & \theta_{\underline{d}}^* &
\end{array}
$$

$$\underline{d} := \max\{\deg(f)/2, d_1, \ldots, d_{m+l}\}$$

Asymptotical convergence and finite Convergence

Under Archimedean's condition, the following hold:

- $\theta_d \uparrow f^*$ and $\theta_d^* \uparrow f^*$ as $d \to \infty$ (Lassere, 2001);
- Finite convergence happens generically (Nie, 2014);
- We can verify global optimality by the so-called rank condition (flat extension/truncation);
- We can easily extract minimizers when the rank condition is satisfied.

Asymptotical convergence and finite Convergence

Under Archimedean's condition, the following hold:

- $\theta_d \uparrow f^*$ and $\theta_d^* \uparrow f^*$ as $d \to \infty$ (Lassere, 2001);
- Finite convergence happens generically (Nie, 2014);
- We can verify global optimality by the so-called rank condition (flat extension/truncation);
- We can easily extract minimizers when the rank condition is satisfied.

Key Message: The moment-SOS hierarchy enables us to approximate/retrieve the global optimum/optimizers via solving a sequence of SDPs with increasing sizes.

Scalability issue

The size of SDP (considering $(\mathrm{Q}_d)^*$) at relaxation order d:

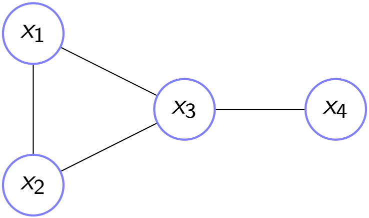
- Maximal size of PSD blocks: $\binom{n+d}{d}$
- Number of equality constraints: $\binom{n+2d}{2d}$

In view of the current state of SDP solvers (e.g. `Mosek`), solvable problems are limited to $n \leq 30$ when $d = 2$ on a standard laptop.

Correlative sparsity (Waki et al., 2006)

Consider $f = x_1^4 + x_1x_2^2 + x_2x_3 + x_3^2x_4^2$ and $g_1 = 1 - x_1^2 - x_2^2 - x_3^2$, $g_2 = 1 - x_3x_4$.

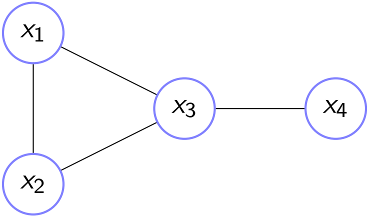
Figure: The correlative sparsity pattern (csp) graph



Correlative sparsity (Waki et al., 2006)

Consider $f = x_1^4 + x_1x_2^2 + x_2x_3 + x_3^2x_4^2$ and $g_1 = 1 - x_1^2 - x_2^2 - x_3^2$, $g_2 = 1 - x_3x_4$.

Figure: The correlative sparsity pattern (csp) graph



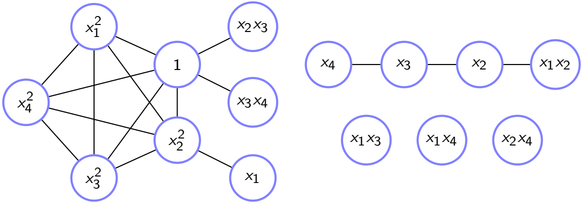
We then construct moment/localizing matrices with respect to the variables involved in each maximal clique of the csp graph:

$$I_k \longmapsto M_d(\mathrm{y}, I_k), M_{d-d_j}(g_j\mathrm{y}, I_k)$$

Term sparsity

Consider $f = x_1^4 + x_1x_2^2 + x_2x_3 + x_3^2x_4^2$ and $g_1 = 1 - x_1^2 - x_2^2 - x_3^2$, $g_2 = 1 - x_3x_4$.

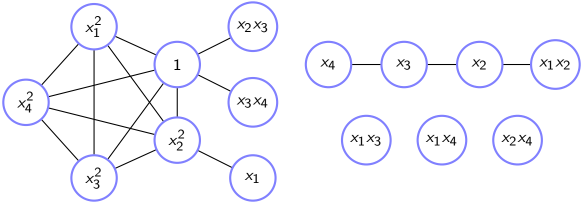
Figure: The term sparsity pattern (tsp) graph with $d = 2$



Term sparsity

Consider $f = x_1^4 + x_1x_2^2 + x_2x_3 + x_3^2x_4^2$ and $g_1 = 1 - x_1^2 - x_2^2 - x_3^2$, $g_2 = 1 - x_3x_4$.

Figure: The term sparsity pattern (tsp) graph with $d = 2$



For each moment/localizing matrix, we iterativly construct an ascending chain of graphs:

$$G_{d,j}^{(1)} \subseteq \cdots \subseteq G_{d,j}^{(k-1)} \subseteq G_{d,j}^{(k)} \subseteq G_{d,j}^{(k+1)} \subseteq \cdots$$

The term sparsity adapted moment-SOS (TSSOS) hierarchy

For each $k \geq 1$ (which is called the sparse order), let us consider

$$\theta_d^{(k)} := \begin{cases} \inf & L_{\mathbf{y}}(f) \\ \text{s.t.} & B_{G_{d,0}^{(k)}} \circ M_d(\mathbf{y}) \in \Pi_{G_{d,0}^{(k)}}(\mathbf{S}_+^{r_0}), \\ & B_{G_{d,j}^{(k)}} \circ M_{d-d_j}(g_j\mathbf{y}) \in \Pi_{G_{d,j}^{(k)}}(\mathbf{S}_+^{r_j}), \quad j = 1, \ldots, m, \\ & B_{G_{d,j}^{(k)}} \circ M_{d-d_j}(g_j\mathbf{y}) = 0, \quad j = m+1, \ldots, m+l, \\ & y_0 = 1. \end{cases}$$

B_G: the adjacency matrix of G

$\Pi_G(\mathbf{S}_+^{|V|})$: the PSD completable cone with specified entries at G

Description of PSD completable cones

Theorem (Grone et al., 1984)

Let $G(V,E)$ be a chordal graph and assume that $C_1,\dots,C_t$ are the list of maximal cliques of $G(V,E)$. Then a matrix $Q \in \Pi_G(\mathbb{S}_+^{|V|})$ if and only if $Q[C_i] \succeq 0$ for $i = 1,\dots,t$, where $Q[C_i]$ denotes the principal submatrix of Q indexed by the clique C_i.

$$\begin{bmatrix} \bullet & \bullet & \bullet & ? \\ \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet \\ ? & \bullet & \bullet & \bullet \end{bmatrix} \succeq 0 \quad \Longleftrightarrow \quad \begin{bmatrix} \bullet & \bullet & \bullet & \\ \bullet & \bullet & \bullet & \\ \bullet & \bullet & \bullet & \\ & & & \end{bmatrix} \succeq 0, \begin{bmatrix} & & & \\ & \bullet & \bullet & \bullet \\ & \bullet & \bullet & \bullet \\ & \bullet & \bullet & \bullet \end{bmatrix} \succeq 0$$

A two-level hierarchy of lower bounds

As a consequence, we obtain a two-level hierarchy of lower bounds for f^*: $(\underline{d} := \max\{\deg(f)/2, d_1, \ldots, d_{m+l}\})$

$$\begin{array}{ccccccc}
\theta^{(1)}_{\underline{d}} & \leq & \theta^{(2)}_{\underline{d}} & \leq & \cdots & \leq & \theta_{\underline{d}} \\
\wedge\!| & & \wedge\!| & & & & \wedge\!| \\
\theta^{(1)}_{\underline{d}+1} & \leq & \theta^{(2)}_{\underline{d}+1} & \leq & \cdots & \leq & \theta_{\underline{d}+1} \\
\wedge\!| & & \wedge\!| & & & & \wedge\!| \\
\vdots & & \vdots & & \vdots & & \vdots \\
\wedge\!| & & \wedge\!| & & & & \wedge\!| \\
\theta^{(1)}_{d} & \leq & \theta^{(2)}_{d} & \leq & \cdots & \leq & \theta_{d} \\
\wedge\!| & & \wedge\!| & & & & \wedge\!| \\
\vdots & & \vdots & & \vdots & & \vdots
\end{array}$$

Combining correlative sparsity with term sparsity (CS-TSSOS)

The combination of correlative sparsity with term sparsity splits into two steps:

1. Partitioning the variables with respect to the maximal cliques of the csp graph;
2. For each subsystem involving variables from one maximal clique, applying the iterative procedure to exploit term sparsity.

$$\begin{cases} \inf & L_{\mathbf{y}}(f) \\ \text{s.t.} & B_{G_{d,l,0}^{(k)}} \circ M_d(\mathbf{y}, I_l) \in \Pi_{G_{d,l,0}^{(k)}}(\mathbf{S}_+^{r_{l,0}}), \quad l = 1, \ldots, p, \\ & B_{G_{d,l,j}^{(k)}} \circ M_{d-d_j}(g_j\mathbf{y}, I_l) \in \Pi_{G_{d,l,j}^{(k)}}(\mathbf{S}_+^{r_{l,j}}), \quad j \in J_l, l = 1, \ldots, p, \\ & B_{G_{d,l,j}^{(k)}} \circ M_{d-d_j}(g_j\mathbf{y}, I_l) = 0, \quad j \in K_l, l = 1, \ldots, p, \\ & y_0 = 1. \end{cases}$$

More techniques

- Chordal extension
  - ▷ maximal chordal extension
  - ▷ approximately smallest chordal extension
- Binary variable ($x^2 = 1$)
- Quotient structure via Gröbner bases
- Extracting an approximate solution and then refining it by `Ipopt`
- Merging PSD blocks

Merging PSD blocks

Figure: Merge two 4×4 blocks into a single 5×5 block

$$\begin{bmatrix} \bullet & \bullet & \bullet & \bullet & \\ \bullet & \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet & \bullet \\ & \bullet & \bullet & \bullet & \bullet \end{bmatrix} \longrightarrow \begin{bmatrix} \bullet & \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet & \bullet \end{bmatrix}$$

Extensions

- Extending to complex polynomial optimization

$$\inf_{\mathbf{z}\in\mathbb{C}^n}\{f(\mathbf{z},\bar{\mathbf{z}}) : g_j(\mathbf{z},\bar{\mathbf{z}}) \geq 0, j=1,\ldots,m, g_j(\mathbf{z},\bar{\mathbf{z}}) = 0, j=m+1,\ldots,m+l\}$$

- Extending to noncommutative polynomial optimization

  ▷ Eigenvalue optimization:

$$\inf_{X}\{\mathrm{eig}\, f(X) : g_j(X) \succeq 0, j=1,\ldots,m, g_j(X) = 0, j=m+1,\ldots,m+l\}$$

  ▷ Trace optimization:

$$\inf_{X}\{\mathrm{tr}\, f(X) : g_j(X) \succeq 0, j=1,\ldots,m, g_j(X) = 0, j=m+1,\ldots,m+l\}$$

TSSOS

Freely available at: https://github.com/wangjie212/TSSOS
Dependency:

- MultivariatePolynomials: manipulate multivariate polynomials
- JuMP: build the SDP problem
- LightGraphs: handle graphs
- MetaGraphs: handle weighted graphs
- ChordalGraph: generate approximately smallest chordal extensions
- SemialgebraicSets: compute Gröbner bases

Besides, TSSOS requires an SDP solver, which can be Mosek, SDPT3, or COSMO.

Usage

```
using TSSOS
using DynamicPolynomials
@polyvar x[1:2]
f = 1 + x[1]^4*x[2]^2 + x[1]^2*x[2]^4 - 3x[1]^2*x[2]^2 # define the
objective function
g = 1 - sum(x[1:2].^2) # define the inequality constraint
pop = [f, g] # define the POP
numeq = 0 # define the number of equality constraints
d = 3 # set a relaxation order
opt,sol,data = tssos_first(pop, x, d, numeq=numeq) # k = 1
opt,sol,data = tssos_higher!(data) # k > 1
opt,sol,data = cs_tssos_first(pop, x, d, numeq=numeq) # k = 1
opt,sol,data = cs_tssos_higher!(data) # k > 1
```

Usage

• For large-scale POPs, it is more efficient to define the supports and coefficients directly.

$$x_1^4 x_2^2 \longrightarrow [1; 1; 1; 1; 2; 2]$$

```
using TSSOS
supp = Vector{Vector{UInt16}}[[[], [1; 1; 1; 1; 2; 2], [1; 1; 2; 2; 2;
2], [1; 1; 2; 2]], [[], [1; 1], [2 ;2]]] # define the support array of
the POP
coe = Vector{Float64}[[1; 1; 1; -3], [1; -1; -1]] # define the
coefficient array of the POP
numeq = 0 # define the number of equality constraints
n = 2 # define the number of variables
d = 3 # set a relaxation order
opt,sol,data = cs_tssos_first(supp, coe, n, d, numeq=numeq) # k = 1
opt,sol,data = cs_tssos_higher!(data) # k > 1
```

Randomly generated polynomials of the SOS form

TSSOS, GloptiPoly, Yalmip: Mosek   SparsePOP: SDPT3

Table: Running time (in seconds) comparison with GloptiPoly, Yalmip and SparsePOP for minimizing randomly generated sparse polynomials of the SOS form giving the same optimum; the symbol "-" indicates out of memory

n	$2d$	TSSOS	GloptiPoly	Yalmip	SparsePOP
8	8	0.24	306	10	24
8	8	0.34	348	13	130
8	8	0.36	326	19	175
8	10	0.58	-	92	323
8	10	0.53	-	72	1526
8	10	0.38	-	22	134
9	10	0.50	-	44	324
9	10	0.72	-	143	-
9	10	0.79	-	109	284
10	12	2.2	-	474	-
10	12	1.6	-	147	318
10	12	1.8	-	350	404
10	16	15	-	-	-
10	16	14	-	-	-
10	16	12	-	-	-
12	12	8.4	-	-	-
12	12	5.7	-	-	-
12	12	7.4	-	-	-

Randomly generated polynomials with simplex Newton polytopes

Table: Running time (in seconds) comparison with GloptiPoly, Yalmip and SparsePOP for minimizing randomly generated sparse polynomials with simplex Newton polytopes giving the same optimum; the symbol "-" indicates out of memory

n	$2d$	TSSOS	GloptiPoly	Yalmip	SparsePOP
8	8	0.36	346	31	271
8	8	0.51	447	24	496
8	8	0.31	257	6.0	178
9	8	1.0	-	-	-
9	8	0.63	-	363	611
9	8	0.76	-	141	578
9	10	6.6	-	322	-
9	10	5.0	-	233	-
9	10	4.9	-	249	-
10	8	1.2	-	-	-
10	8	8.0	-	536	-
10	8	1.0	-	-	-
11	8	1.7	-	655	398
11	8	1.8	-	-	221
11	8	1.9	-	340	293
12	8	10	-	-	-
12	8	7.4	-	-	-
12	8	2.9	-	-	-

The AC optimal power flow problem

Table: The results for AC-OPF instances; mb: the maximal size of blocks, gap: the relative gap with respect to a local optimal solution, -: out of memory

n	m+l	CS ($d=2$)				CS+TS ($d=2$, $k=1$)			
		mb	opt	time (s)	gap	mb	opt	time (s)	gap
12	28	28	1.1242e4	0.21	0.00%	22	1.1242e4	0.09	0.00%
20	55	28	1.7543e4	0.56	0.05%	22	1.7543e4	0.30	0.05%
114	315	66	1.3442e5	5.59	0.39%	31	1.3396e5	2.01	0.73%
114	315	120	7.6943e4	94.9	0.00%	39	7.6942e4	14.8	0.00%
72	297	45	4.9927e3	4.43	0.07%	22	4.9920e3	2.69	0.08%
344	971	153	4.2246e5	758	0.06%	44	4.2072e5	96.0	0.48%
344	971	153	2.2775e5	504	0.00%	44	2.2766e5	71.5	0.04%
344	1325	253	–	–	–	31	2.4180e5	82.7	0.11%
344	1325	253	–	–	–	73	1.0470e5	169	0.50%
348	1809	253	–	–	–	34	1.0802e5	278	0.05%
348	1809	253	–	–	–	34	1.2096e5	201	0.03%
766	3322	153	3.3072e6	585	0.68%	44	3.3042e6	33.9	0.77%
1112	4613	231	4.2413e4	3114	0.85%	39	4.2408e4	46.6	0.86%
1112	4613	496	–	–	–	31	7.2396e4	410	0.25%
4356	18257	378	–	–	–	27	1.3953e6	934	0.51%
6698	29283	1326	–	–	–	76	5.9858e5	1886	0.47%

Eigenvalue minimization of the nc generalized Rosenbrock function

Table: The eigenvalue minimization of the nc generalized Rosenbrock function over $\mathcal{D}$, where $\mathcal{D}$ is defined by $\{1 - X_1^2 \succeq 0, \ldots, 1 - X_n^2 \succeq 0, X_1 - 1/3 \succeq 0, \ldots, X_n - 1/3 \succeq 0\}$; the symbol "-" indicates out of memory

n	Sparse ($d = 2$, $k = 1$)			Dense ($d = 2$)		
	mb	opt	time (s)	mb	opt	time (s)
20	3	1.0000	0.14	-	-	-
40	3	1.0000	0.22	-	-	-
60	3	0.9999	0.28	-	-	-
80	3	0.9999	0.35	-	-	-
100	3	0.9999	0.46	-	-	-
200	3	0.9999	0.89	-	-	-
400	3	1.0000	2.40	-	-	-
600	3	1.0000	4.47	-	-	-
800	3	1.0000	6.95	-	-	-
1000	3	0.9999	10.2	-	-	-
2000	3	0.9999	37.2	-	-	-
3000	3	0.9999	87.2	-	-	-
4000	3	0.9998	145	-	-	-

Trace minimization of the nc Broyden tridiagonal function

Table: The trace minimization of the nc Broyden tridiagonal function over $\mathcal{D}$, where $\mathcal{D}$ is defined by $\{1 - X_1^2 \succeq 0, \ldots, 1 - X_n^2 \succeq 0, X_1 - 1/3 \succeq 0, \ldots, X_n - 1/3 \succeq 0\}$; the symbol "-" indicates out of memory

n	Sparse ($d = 2$, $k = 1$)			Dense ($d = 2$)		
	mb	opt	time (s)	mb	opt	time (s)
20	6	1.1805	0.27	-	-	-
40	6	1.1828	0.53	-	-	-
60	6	1.1828	0.68	-	-	-
80	6	1.1828	0.82	-	-	-
100	6	1.1828	1.07	-	-	-
200	6	1.1828	2.45	-	-	-
400	6	1.1828	6.18	-	-	-
600	6	1.1828	12.2	-	-	-
800	6	1.1828	20.1	-	-	-
1000	6	1.1828	28.6	-	-	-
2000	6	1.1828	104	-	-	-
3000	6	1.1828	204	-	-	-
4000	6	1.1828	363	-	-	-

Main references

- Jie Wang, Victor Magron and Jean B. Lasserre, *TSSOS: A Moment-SOS hierarchy that exploits term sparsity*, SIAM Optimization, 2021.
- Jie Wang, Victor Magron and Jean B. Lasserre, *Chordal-TSSOS: a moment-SOS hierarchy that exploits term sparsity with chordal extension*, SIAM Optimization, 2021.
- Jie Wang, Victor Magron, Jean B. Lasserre and Ngoc H. A. Mai, *CS-TSSOS: Correlative and term sparsity for large-scale polynomial optimization*, arXiv:2005.02828, 2020.
- Jie Wang and Victor Magron, *Exploiting Term Sparsity in Noncommutative Polynomial Optimization*, arXiv:2010.06956, 2021.
- Jie Wang and Victor Magron, *Exploiting Sparsity in Complex Polynomial Optimization*, arXiv:2103.12444, 2021.
- `TSSOS`: https://github.com/wangjie212/TSSOS

Thanks for your attention!