TSSOS: A JULIA LIBRARY TO EXPLOIT SPARSITY FOR LARGE-SCALE POLYNOMIAL OPTIMIZATION

VICTOR MAGRON AND JIE WANG

Abstract. The Julia library `TSSOS` aims at helping polynomial optimizers to solve large-scale problems with sparse input data. The underlying algorithmic framework is based on exploiting correlative and term sparsity to obtain a new moment-SOS hierarchy involving potentially much smaller positive semidefinite matrices. `TSSOS` can be applied to numerous problems ranging from power networks to eigenvalue and trace optimization of noncommutative polynomials, involving up to tens of thousands of variables and constraints.

1. Introduction

The `TSSOS` library intends to address large-scale polynomial optimization problems, where the polynomials in the problem's description involve only a small number of terms compared to the dense ones.

The library is available at https://github.com/wangjie212/TSSOS. The ultimate goal is to provide semidefinite relaxations that are computationally much cheaper than those of the standard SOS-based hierarchy [11] or its sparse version [12, 21] based on correlative sparsity by taking into account the so-called *term sparsity*. Existing algorithmic frameworks based on correlative sparsity have been implemented in the `SparsePOP` solver [22] and many applications of interest have been successfully handled, for instance certified roundoff error bounds [13, 14], optimal powerflow problems [9], volume computation [19], dynamical systems [18], noncommutative POPs [10], Lipschitz constant estimation of deep networks [5, 6] and sparse positive definite functions [17].

Throughout the paper, we consider the following formulation of the polynomial optimization problem (POP):

$$\text{(Q)}: \quad \rho^* = \inf_{\mathbf{x}} \{ f(\mathbf{x}) : \mathbf{x} \in \mathbf{K} \}, \tag{1.1}$$

where the objective function f is assumed to be a polynomial in n variables $\mathbf{x} = (x_1, \ldots, x_n)$ and the feasible region $\mathbf{K} \subseteq \mathbb{R}^n$ is assumed to be defined by a finite conjunction of m polynomial inequalities and t polynomial equalities, namely

$$\mathbf{K} := \{\mathbf{x} \in \mathbb{R}^n : g_1(\mathbf{x}) \geq 0, \ldots, g_m(\mathbf{x}) \geq 0, h_1(\mathbf{x}) = 0, \ldots, h_t(\mathbf{x}) = 0\}, \tag{1.2}$$

for some polynomials $g_1, \ldots, g_m, h_1, \ldots, h_t$ in $\mathbf{x}$. For the sake of simplicity, we assume in what follows that there are only inequalities in (1.2), i.e., $t = 0$. A nowadays well-established scheme to handle (Q) is the *moment-SOS hierarchy* [11], where SOS is the abbreviation of *sum of squares*. The moment-SOS hierarchy

Date: March 1, 2021.

2010 *Mathematics Subject Classification.* Primary, 14P10,90C25; Secondary, 12D15,12Y05.

Key words and phrases. moment-SOS hierarchy, correlative sparsity, term sparsity, TSSOS, large-scale (non)-commutative polynomial optimization, optimal power flow.

provides a sequence of semidefinite programming (SDP) relaxations, whose optimal values are non-decreasing lower bounds of the global optimal value ρ^* of (Q). Under certain mild conditions, the sequence of lower bounds is guaranteed to converge to ρ^* generically in finite many steps. Despite of this beautiful theoretically property, the bottleneck of the scheme is that the size of the SDP relaxations become quickly intractable as n increases. Hence in order to improve the scalability, it is crucial to fully exploit the structure of POP (1.1) to reduce the size of these relaxations. A commonly present structure in large-scale polynomial optimization is *sparsity*. In view of this, `TSSOS` implements the sparsity-adapted moment-SOS hierarchies. Here the terminology "sparsity" is referred to the well-known correlative sparsity, or the newly proposed term sparsity [26, 27], or the combined both [28]. The idea of exploitation of sparsity behind `TSSOS` is not restricted to POPs, but can be also applied to other SOS optimization problems, e.g., the computation of joint spectral radii [24] or learning of linear dynamical systems [29, 30]. `TSSOS` can be also combined with efficient first-order methods to speed up the computation of the SDP relaxations themselves [15, 16].

2. Algorithmic background and overall description

Let $d_j = \lceil \deg(g_j)/2 \rceil, j = 1, \ldots, m$ and $d_{\min} = \max\{\lceil \deg(f)/2 \rceil, d_1, \ldots, d_m\}$. The moment hierarchy indexed by the integer $d \geq d_{\min}$ for POP (1.1) is defined by:

$$(\mathrm{Q}_d): \quad \begin{cases} \inf & L_{\mathbf{y}}(f) \\ \text{s.t.} & \mathbf{M}_d(\mathbf{y}) \succeq 0, \\ & \mathbf{M}_{d-d_j}(g_j\mathbf{y}) \succeq 0, \quad j \in [m], \\ & y_{\mathbf{0}} = 1. \end{cases} \tag{2.1}$$

Here $\mathbf{M}_d(\mathbf{y})$ is the d-th order moment matrix, $\mathbf{M}_{d-d_j}(g_j\mathbf{y})$ is the $(d-d_j)$-th order localizing matrix (see [11] for more details) and $[m] := \{1, \ldots, m\}$ for a positive integer m. The index d is called the *relaxation order* of the hierarchy. Note that if $d < d_{\min}$, then (Q_d) is infeasible.

2.1. **Correlative sparsity.** By exploiting correlative sparsity, we decompose the set of variables into subsets and then construct moment (localizing) matrices for each subset. Fix a relaxation order $d \geq d_{\min}$. Let $J' := \{j \in [m] \mid d_j = d\}$. For a polynomial $h = \sum_{\boldsymbol{\alpha}} h_{\boldsymbol{\alpha}} \mathbf{x}^{\boldsymbol{\alpha}}$ ($\mathbf{x}^{\boldsymbol{\alpha}} := x_1^{\alpha_1} \cdots x_n^{\alpha_n}$), the *support* of h is defined by $\operatorname{supp}(h) := \{\boldsymbol{\alpha} \in \mathbb{N}^n \mid h_{\boldsymbol{\alpha}} \neq 0\}$. For $\boldsymbol{\alpha} = (\alpha_i) \in \mathbb{N}^n$, let $\operatorname{supp}(\boldsymbol{\alpha}) := \{i \in [n] \mid \alpha_i \neq 0\}$. The *correlative sparsity pattern (csp) graph* associated with POP (1.1) is defined to be the graph G^{csp} with nodes $V = [n]$ and edges E satisfying $\{i, j\} \in E$ if one of followings holds:

(i) there exists $\boldsymbol{\alpha} \in \operatorname{supp}(f) \cup \bigcup_{j \in J'} \operatorname{supp}(g_j)$ such that $\{i, j\} \subseteq \operatorname{supp}(\boldsymbol{\alpha})$;
(ii) there exists $k \in [m] \backslash J'$ such that $\{i, j\} \subseteq \bigcup_{\boldsymbol{\alpha} \in \operatorname{supp}(g_k)} \operatorname{supp}(\boldsymbol{\alpha})$.

Let $\overline{G}^{\mathrm{csp}}$ be a chordal extension of G^{csp} and $\{I_l\}_{l \in [p]}$ be the list of maximal cliques of $\overline{G}^{\mathrm{csp}}$. The polynomials $g_j, j \in [m] \backslash J'$ can be then partitioned into groups $\{g_j \mid j \in J_l\}, l \in [p]$ with respect to variable subsets $\{\mathbf{x}(I_l)\}_{l \in [p]}$, where $\mathbf{x}(I_l) := \{x_i \mid i \in I_l\}$. Consequently, we obtain the following moment hierarchy indexed by

d based on correlative sparsity:

$$(\mathrm{Q}_d^{\mathrm{cs}}): \quad \begin{cases} \inf & L_{\mathbf{y}}(f) \\ \text{s.t.} & \mathbf{M}_d(\mathbf{y}, I_l) \succeq 0, \quad l \in [p], \\ & \mathbf{M}_{d-d_j}(g_j\mathbf{y}, I_l) \succeq 0, \quad j \in J_l, l \in [p], \\ & L_{\mathbf{y}}(g_j) \geq 0, \quad j \in J', \\ & y_{\mathbf{0}} = 1. \end{cases}$$

Here $\mathbf{M}_d(\mathbf{y}, I_l)$ ($\mathbf{M}_{d-d_j}(g_j\mathbf{y})$) is the moment (localizing) matrix constructed with respect to the variables $\mathbf{x}(I_l)$.

2.2. **Term sparsity.** By exploiting term sparsity, we are able to construct moment (localizing) matrices with a block structure in an iterative manner. For $d \in \mathbb{N}$, let $\mathbb{N}^n_d := \{\boldsymbol{\alpha} = (\alpha_i) \in \mathbb{N}^n \mid \sum_{i=1}^n \alpha_i \leq d\}$. Fix a relaxation order $d \geq d_{\min}$. Set $d_0 := 0$ and $g_0 := 1$. Let $\mathscr{S}_0 = \mathrm{supp}(f) \cup \bigcup_{j=1}^m \mathrm{supp}(g_j) \cup (2\mathbb{N}^n_d)$ be the initial support. For each step $k \geq 1$, let $F^{(k)}_{d,j}$ be the graph with $V(F^{(k)}_{d,j}) = \mathbb{N}^n_{d-d_j}$ and

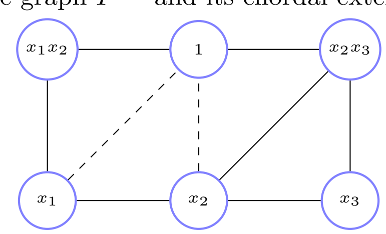
$$E(F^{(k)}_{d,j}) = \{\{\boldsymbol{\beta}, \boldsymbol{\gamma}\} \mid (\boldsymbol{\beta} + \boldsymbol{\gamma} + \mathrm{supp}(g_j)) \cap \mathscr{S}_{k-1} \neq \emptyset\} \tag{2.2}$$

for $j \in \{0\} \cup [m]$ and let $G^{(k)}_{d,j}$ be a chordal extension of $F^{(k)}_{d,j}$. The extended support at the k-th step is defined as

$$\mathscr{S}_k := \bigcup_{i=0}^m \{\boldsymbol{\alpha} + \boldsymbol{\beta} + \boldsymbol{\gamma} \mid \boldsymbol{\alpha} \in \mathrm{supp}(g_j), \{\boldsymbol{\beta}, \boldsymbol{\gamma}\} \in E(G^{(k-1)}_{d,j}) \text{ or } \boldsymbol{\beta} = \boldsymbol{\gamma}\}.$$

Example 2.1. *Consider the polynomial* $f = 1 + x_1^2 + x_1x_2 + x_2^2 + x_1^2x_2 + x_1^2x_2^2 + x_2x_3 + x_3^2 + x_2^2x_3 + x_2x_3^2 + x_2^2x_3^2$. *To minimize* f *over* $\mathbb{R}^3$, *we can take the monomial basis* $\{1, x_1, x_2, x_3, x_1x_2, x_2x_3\}$. *Figure 1 shows the graph* $F^{(1)}$ *(without dashed edges) and its chordal extension* $G^{(1)}$ *(with dashed edges) for* f, *where we omit the subscripts* d, j *since there is no constraint.*

FIGURE 1. The graph $F^{(1)}$ and its chordal extension $G^{(1)}$ for f



Let $\{C^{(k)}_{d,j,i}\}_{i=1}^{s_{d,j}}$ be the list of maximal cliques of $G^{(k)}_{d,j}$ for $j = 0, \ldots, m$. Then the moment hierarchy based on term sparsity for POP (1.1) is defined as:

$$(\mathrm{Q}^{\mathrm{ts}}_{d,k}): \quad \begin{cases} \inf & L_{\mathbf{y}}(f) \\ \text{s.t.} & [\mathbf{M}_d(\mathbf{y})]_{C^{(k)}_{d,0,i}} \succeq 0, \quad i \in [s_{d,0}], \\ & [\mathbf{M}_{d-d_j}(g_j\mathbf{y})]_{C^{(k)}_{d,j,i}} \succeq 0, \quad i \in [s_{d,j}], j \in [m], \\ & y_{\mathbf{0}} = 1. \end{cases} \tag{2.3}$$

Here we denote by A_C the submatrix of $A \in \mathbb{R}^{r\times r}$ with rows and columns indexed by $C \subseteq [r]$.

The above hierarchy (called the TSSOS hierarchy) is indexed by two parameters: the relaxation order d and the *sparse order* k. For a fixed d, the sequence of optimums of $(\mathrm{Q}^{\mathrm{ts}}_{d,k})$ is non-decreasing and stabilizes in finitely many steps. There are two particular choices for the chordal extension $G^{(k)}_{d,j}$ of $F^{(k)}_{d,j}$: approximately smallest chordal extensions [26][1] and the maximal chordal extension [27][2]. This offers a trade-off between the computational cost and the quality of obtained lower bounds. Typically, the choice of approximately smallest chordal extensions leads to positive semidefinite (PSD) blocks of smaller sizes but may provide a looser lower bound whereas the choice of the maximal chordal extension leads to PSD blocks of larger sizes but may provide a tighter lower bound. It was proved in [27] that if the maximal chordal extension is chosen, then the sequence of optimums of $(\mathrm{Q}^{\mathrm{ts}}_{d,k})$ converges to the optimum of the corresponding dense relaxation as k increases for a fixed relaxation order d.

2.3. **Correlative-term sparsity.** We can further exploit correlative sparsity and term sparsity simultaneously. Namely, first partition the set of variables into subsets $\{I_l\}_{l\in[p]}$ as done in Sec. 2.1 and then apply the iterative procedure for exploiting term sparsity in Sec. 2.2 to each subsystem involving variables $\mathbf{x}(I_l)$. Let $G^{(k)}_{d,l,j}$ be the graphs associated with each subsystem and $\{C^{(k)}_{d,l,j,i}\}_{i=1}^{s_{d,l,j}}$ be the list of maximal cliques of $G^{(k)}_{d,l,j}$ for $j \in \{0\} \cup J_l, l \in [p]$. Then the moment hierarchy based on correlative-term sparsity for POP (1.1) is defined as:

$$(2.4)\quad (\mathrm{Q}^{\text{cs-ts}}_{d,k}):\quad \begin{cases} \inf & L_{\mathbf{y}}(f) \\ \text{s.t.} & [\mathbf{M}_d(\mathbf{y}, I_l)]_{C^{(k)}_{d,l,0,i}} \succeq 0, \quad i \in [s_{d,l,0}], l \in [p], \\ & [\mathbf{M}_{d-d_j}(g_j\mathbf{y}, I_l)]_{C^{(k)}_{d,l,j,i}} \succeq 0, \quad i \in [s_{d,l,j}], j \in J_l, l \in [p], \\ & L_{\mathbf{y}}(g_j) \ge 0, \quad j \in J', \\ & y_{\mathbf{0}} = 1. \end{cases}$$

The above hierarchy (called the CS-TSSOS hierarchy) is also indexed by two parameters: the relaxation order d and the *sparse order* k. For a fixed d, the sequence of optimums of $(\mathrm{Q}^{\text{cs-ts}}_{d,k})$ is non-decreasing and stabilizes in finitely many steps. As discussed in Sec. 2.2, the choice of chordal extensions $G^{(k)}_{d,l,j}$ offers a trade-off between the computational cost and the quality of obtained lower bounds. It was proved in [28] that if the maximal chordal extension is chosen, then the sequence of optimums of $(\mathrm{Q}^{\text{cs-ts}}_{d,k})$ converges to the optimum of $(\mathrm{Q}^{\mathrm{cs}}_{d})$ as k increases for a fixed relaxation order d.

[1] A smallest chordal extension is a chordal extension with the smallest clique number. Computing a smallest chordal extension is an NP-hard problem. Fortunately, there are efficient heuristic algorithms to produce a good approximation of smallest chordal extensions.

[2] By the maximal chordal extension, we refer to the chordal extension that completes each connected component of the graph.

3. The implementation of TSSOS

TSSOS, which implements the sparsity-adapted moment-SOS hierarchies (i.e. $(\mathrm{Q}_d^{\mathrm{cs}})$, $(\mathrm{Q}_{d,k}^{\mathrm{ts}})$ and $(\mathrm{Q}_{d,k}^{\text{cs-ts}})$), is developed as a Julia library, aiming to solve large-scale polynomial optimization problems by fully exploiting sparsity as well as other techniques. TSSOS provides an easy way to define a POP and to solve it by SDP relaxations. The tunable parameters (e.g. d, k, the types of chordal extensions) allow the user to find the best compromise between the computational cost and the solving precision. The following script is a simple example to illustrate the usage of TSSOS.

```
using TSSOS
using DynamicPolynomials
@polyvar x[1:6]
f = x[1]^4 + x[2]^4 - 2x[1]^2*x[2] - 2x[1] + 2x[2]*x[3] - 2x[1]^2*x[3]
- 2x[2]^2*x[3] - 2x[2]^2*x[4] - 2x[2] + 2x[1]^2 + 2.5x[1]*x[2] - 2x[4]
+ 2x[1]*x[4] + 3x[2]^2 + 2x[2]*x[5] + 2x[3]^2 + 2x[3]*x[4] + 2x[4]^2 +
x[5]^2 - 2x[5] + 2 # define the objective function
g = 1 - sum(x[1:2].^2) # define the inequality constraint
h = 1 - sum(x[3:5].^2) # define the equality constraint
d = 2 # define the relaxation order
numeq = 1 # define the number of equality constraints
```

To solve the first step of the TSSOS hierarchy with approximately smallest chordal extensions (option TS="MD"), run

```
opt,sol,data = tssos_first([f;g;h], x, d, numeq=numeq, TS="MD")
```

We obtain

```
optimum = 0.20967292920706904
```

To solve higher steps of the TSSOS hierarchy, repeatedly run

```
opt,sol,data = tssos_higher!(data, TS="MD")
```

For instance, for the second step of the TSSOS hierarchy we obtain

```
optimum = 0.21230011405774876
```

To solve the first step of the CS-TSSOS hierarchy, run

```
opt,sol,data = cs_tssos_first([f;g;h], x, d, numeq=numeq, TS="MD")
```

We obtain

```
optimum = 0.20929635879961658
```

To solve higher steps of the CS-TSSOS hierarchy, repeatedly run

```
opt,sol,data = cs_tssos_higher!(data, TS="MD")
```

For instance, for the second step of the CS-TSSOS hierarchy we obtain

```
optimum = 0.20974835386107363
```

3.1. **Dependencies.** TSSOS depends on the following Julia packages:

- MultivariatePolynomials to manipulate multivariate polynomials;
- JuMP [7] to model the SDP problem;
- LightGraphs [4] to handle graphs;
- MetaGraphs to handle weighted graphs;
- ChordalGraph [23] to generate approximately smallest chordal extensions;

- `SemialgebraicSets` to compute Gröbner bases.

Besides, `TSSOS` requires an SDP solver, which can be `MOSEK` [1], `SDPT3` [20], or `COSMO` [8]. Once one of the SDP solvers has been installed, the installation of `TSSOS` is straightforward:

```
Pkg.add("https://github.com/wangjie212/TSSOS")
```

3.2. **Binary variables.** `TSSOS` supports binary variables. By setting `nb` $= s$, one can specify that the first s variables are binary variables $x_1, \ldots, x_s$ which satisfy the equation $x_i^2 = 1$, $i \in [s]$. The specification is helpful to reduce the number of decision variables of SDP relaxations since one can identify x^r with $x^{r \,(\mathrm{mod}\ 2)}$ for a binary variable x.

3.3. **Equality constraints.** If there are equality constraints in the description of POP (1.1), then one can reduce the number of decision variables of SDP relaxations by working in the quotient ring $\mathbb{R}[\mathbf{x}]/(h_1, \ldots, h_t)$, where $\{h_1 = 0, \ldots, h_t = 0\}$ is the set of equality constraints. To conduct the elimination, we need to compute a Gröbner basis GB of the ideal $(h_1, \ldots, h_t)$. Then any monomial $\mathbf{x}^{\boldsymbol{\alpha}}$ can be replaced by its normal form $\mathrm{NF}(\mathbf{x}^{\boldsymbol{\alpha}}, GB)$ with respect to the Gröbner basis GB when constructing SDP relaxations. This reduction is conducted by default for the TSSOS hierarchy in `TSSOS`.

3.4. **Adding extra first-order moment matrices.** When POP (1.1) is a quadratically constrained quadratic program, the first-order moment-SOS relaxation is also known as Shor's relaxation. In this case, (Q_1), $(\mathrm{Q}_1^{\mathrm{cs}})$ and $(\mathrm{Q}_{1,1}^{\mathrm{ts}})$ yield the same optimum. To ensure that any higher order sparse relaxation (i.e. $(\mathrm{Q}_{d,k}^{\text{cs-ts}})$ with $d > 1$) provides a tighter lower bound compared to the one given by Shor's relaxation, we may add an extra first-order moment matrix for each variable clique in $(\mathrm{Q}_{d,k}^{\text{cs-ts}})$:

$$
(3.1) \quad (\mathrm{Q}_{d,k}^{\text{cs-ts}})' : \quad \begin{cases} \inf & L_{\mathbf{y}}(f) \\ \text{s.t.} & [\mathbf{M}_d(\mathbf{y}, I_l)]_{C_{d,l,0,i}^{(k)}} \succeq 0, \quad i \in [s_{d,l,0}], l \in [p], \\ & [\mathbf{M}_1(\mathbf{y}, I_l)] \succeq 0, \quad l \in [p], \\ & [\mathbf{M}_{d-d_j}(g_j \mathbf{y}, I_l)]_{C_{d,l,j,i}^{(k)}} \succeq 0, \quad i \in [s_{d,l,j}], j \in J_l, l \in [p], \\ & L_{\mathbf{y}}(g_j) \geq 0, \quad j \in J', \\ & y_{\mathbf{0}} = 1. \end{cases}
$$

In `TSSOS`, this is accomplished by setting `MomentOne` $=$ `true`.

3.5. **Chordal extensions.** For correlative sparsity, `TSSOS` uses approximately smallest chordal extensions. For term sparsity, `TSSOS` supports two types of chordal extensions: the maximal chordal extension (option `TS` = "`block`") and approximately smallest chordal extensions. `TSSOS` generates approximately smallest chordal extensions via two heuristics: the Minimum Degree heuristic (option "`TS` = `MD`") and the Minimum Fillin heuristic (option "`TS` = `MF`"). See [3] for a full description of these two heuristics. The Minimum Degree heuristic is slightly faster in practice, but the Minimum Fillin heuristic yields on average slightly smaller clique numbers. Hence for correlative sparsity, the Minimum Degree heuristic is recommended and for term sparsity, the Minimum Fillin heuristic is recommended.

FIGURE 2. Merge two 4×4 blocks into a single 5×5 block

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

3.6. **Merging PSD blocks.** In case that two PSD blocks have a large portion of overlaps, it might be beneficial to merge these two blocks into a single block for efficiency. See Figure 2 for such an example. `TSSOS` supports PSD block merging inspired by the strategy proposed in [8]. To activate the merging process, one just needs to set the option `Merge` = `True`. The parameter `md` = `3` can be used to tune the merging strength.

3.7. **Representing polynomials in terms of supports and coefficients.** The Julia package `DynamicPolynomials` provides an efficient way to define polynomials symbolically. But for large-scale polynomial optimization (say, $n > 500$), it is more efficient to represent polynomials by their supports and coefficients. For instance, we can represent $f = x_1^4 + x_2^4 + x_3^4 + x_1x_2x_3$ in terms of its support and coefficients as follows:

```
supp = [[1; 1; 1; 1], [2; 2; 2 ;2], [3; 3; 3; 3], [1; 2; 3]] # define
the support array of f
coe = [1; 1; 1; 1] # define the coefficient vector of f
```

The above representation of polynomials is natively supported by `TSSOS`. Hence the user can define the polynomial optimization problem directly by the support data and the coefficient data to speed up the modeling process.

3.8. **Extension to noncommutative polynomial optimization.** The whole framework of exploiting sparsity for (commutative) polynomial optimization can be extended to handle noncommutative polynomial optimization [25], including eigenvalue and trace optimization, which leads to the submodule `NCTSSOS` in `TSSOS`. Table 1 displays the numerical results for the eigenvalue minimization of the noncommutative Broyden banded function. Here "mb" stands for the maximal block size of the matrices involved in the SDP relaxation. It is evident that the sparse approach scales much better than the dense one.

4. NUMERICAL EXPERIMENTS

In this section, we present numerical results for the alternating current optimal power flow (AC-OPF) problem – a famous industrial problem in power system, which can be cast as a POP involving up to tens of thousands of variables and constraints. To tackle an AC-OPF instance, we first compute a locally optimal solution with a local solver and then rely on the sparsity-adapted moment-SOS hierarchy to certify the global optimality. Suppose that the optimal value reported by the local solver is AC and the optimal value of the SDP relaxation is opt. Then the *optimality gap* between the locally optimal solution and the SDP relaxation is

Table 1. The eigenvalue minimization of the noncommutative Broyden banded function: exploiting sparsity versus without exploiting sparsity. n: the number of variables; mb: the maixmal size of PSD blocks; opt: the optimum returned by the SDP solver; time: running time in seconds; "-" indicates an out of memory error.

n	sparse			dense		
	mb	opt	time	mb	opt	time
20	15	0	0.18	61	0	1.39
40	15	0	0.72	121	0	66.1
60	15	0	1.05	181	0	505
80	15	0	1.24	-	-	-
100	15	0	1.41	-	-	-
200	15	0	3.25	-	-	-
400	15	-0.0001	6.70	-	-	-
600	15	-0.0002	13.2	-	-	-

defined by

$$\text{gap} := \frac{\text{AC} - \text{opt}}{\text{AC}} \times 100\%.$$

When the optimality gap is less than 1%, we accept the locally optimal solution as globally optimal. The test cases in this section are selected from the AC-OPF library *PGLiB* [2]. All numerical experiments were computed on an Intel Core i5-8265U@1.60GHz CPU with 8GB RAM memory. The SDP solver is `Mosek` 9.0 with default parameters.

Table 2. The results for AC-OPF instances. n: the number of variables; m: the number of constraints; AC: the local optimum; mc: the maximal size of variable cliques; opt: the optimum returned by the SDP solver; time: running time in seconds; gap: the optimality gap.

case	n	m	AC	mc	opt	time	gap
14_ieee_api	38	147	5.9994e3	6	5.9994e3	0.54	0.00%
30_ieee	72	297	8.2085e3	8	8.2085e3	0.99	0.00%
39_epri_sad	98	361	1.4834e5	8	1.4831e5	1.45	0.02%
118_ieee	344	1325	9.7214e4	10	9.7214e4	7.71	0.00%
179_goc_api	416	1827	1.9320e6	10	1.9226e6	9.69	0.48%
300_ieee	738	2983	5.6522e5	14	5.6522e5	25.2	0.00%
793_goc	1780	7019	2.6020e5	18	2.5932e5	66.1	0.34%
1354_pegase_sad	3228	13901	1.2588e6	26	1.2582e6	387	0.05%
1951_rte_api	4634	18921	2.4108e6	26	2.4029e6	596	0.32%
2000_goc_api	4476	23009	1.4686e6	42	1.4610e6	1094	0.51%
2312_goc	5076	21753	4.4133e5	68	4.3858e5	997	0.62%
3022_goc_sad	6698	29283	6.0143e5	50	5.9859e5	1340	0.47%

From Table 2, we can see that for all test cases, TSSOS successfully reduces the optimality gap to less than 1%, namely, certifies the global optimality. The largest instance 3022_goc_sad has 6698 variables and 29283 constraints.

Acknowledgements. This work was supported by the Tremplin ERC Stg Grant ANR-18-ERC2-0004-01 (T-COPS project). The firts author was supported by the FMJH Program PGMO (EPICS project) and EDF, Thales, Orange et Criteo. This work has benefited from the European Union's Horizon 2020 research and innovation program under the Marie Sklodowska-Curie Actions, grant agreement 813211 (POEMA) as well as from the AI Interdisciplinary Institute ANITI funding, through the French "Investing for the Future PIA3" program under the Grant agreement n°ANR-19-PI3A-0004.

References

[1] MOSEK ApS. *The MOSEK optimization toolbox. Version 8.1.*, 2017.
[2] Sogol Babaeinejadsarookolaee, Adam Birchfield, Richard D Christie, Carleton Coffrin, Christopher DeMarco, Ruisheng Diao, Michael Ferris, Stephane Fliscounakis, Scott Greene, Renke Huang, et al. The power grid library for benchmarking ac optimal power flow algorithms. *arXiv preprint arXiv:1908.02788*, 2019.
[3] Hans L Bodlaender and Arie MCA Koster. Treewidth computations i. upper bounds. *Information and Computation*, 208(3):259–275, 2010.
[4] Seth Bromberger, James Fairbanks, and other contributors. Juliagraphs/lightgraphs.jl: an optimized graphs package for the julia programming language, 2017.
[5] Tong Chen, Jean-Bernard Lasserre, Victor Magron, and Edouard Pauwels. Semialgebraic optimization for lipschitz constants of relu networks. *arXiv e-prints*, pages arXiv–2002, 2020.
[6] Tong Chen, Jean-Bernard Lasserre, Victor Magron, and Edouard Pauwels. A sublevel moment-sos hierarchy for polynomial optimization. *arXiv preprint arXiv:2101.05167*, 2021.
[7] Iain Dunning, Joey Huchette, and Miles Lubin. Jump: A modeling language for mathematical optimization. *SIAM Review*, 59(2):295–320, 2017.
[8] Michael Garstka, Mark Cannon, and Paul Goulart. Cosmo: A conic operator splitting method for large convex problems. In *2019 18th European Control Conference (ECC)*, pages 1951–1956. IEEE, 2019.
[9] Cédric Josz and Daniel K Molzahn. Lasserre hierarchy for large scale polynomial optimization in real and complex variables. *SIAM Journal on Optimization*, 28(2):1017–1048, 2018.
[10] Igor Klep, Victor Magron, and Janez Povh. Sparse noncommutative polynomial optimization. *Mathematical Programming*, pages 1–41, 2021.
[11] J.-B. Lasserre. Global Optimization with Polynomials and the Problem of Moments. *SIAM Journal on Optimization*, 11(3):796–817, 2001.
[12] J.-B. Lasserre. Convergent sdp-relaxations in polynomial optimization with sparsity. *SIAM Journal on Optimization*, 17(3):822–843, 2006.
[13] Victor Magron. Interval enclosures of upper bounds of roundoff errors using semidefinite programming. *ACM Transactions on Mathematical Software (TOMS)*, 44(4):1–18, 2018.
[14] Victor Magron, George Constantinides, and Alastair Donaldson. Certified roundoff error bounds using semidefinite programming. *ACM Transactions on Mathematical Software (TOMS)*, 43(4):1–31, 2017.
[15] Ngoc Hoang Anh Mai, Abhishek Bhardwaj, and Victor Magron. The constant trace property in noncommutative optimization. *arXiv preprint arXiv:2102.02162*, 2021.
[16] Ngoc Hoang Anh Mai, Jean-Bernard Lasserre, Victor Magron, and Jie Wang. Exploiting constant trace property in large-scale polynomial optimization. *arXiv preprint arXiv:2012.08873*, 2020.
[17] Ngoc Hoang Anh Mai, Victor Magron, and J-B Lasserre. A sparse version of reznick's positivstellensatz. *arXiv preprint arXiv:2002.05101*, 2020.
[18] Corbinian Schlosser and Milan Korda. Sparse moment-sum-of-squares relaxations for nonlinear dynamical systems with guaranteed convergence. *arXiv preprint arXiv:2012.05572*, 2020.

[19] Matteo Tacchi, Tillmann Weisser, Jean-Bernard Lasserre, and Didier Henrion. Exploiting sparsity for semi-algebraic set volume computation. *arXiv preprint arXiv:1902.02976*, 2019.
[20] Kim-Chuan Toh, Michael J Todd, and Reha H Tütüncü. Sdpt3—a matlab software package for semidefinite programming, version 1.3. *Optimization methods and software*, 11(1-4):545–581, 1999.
[21] H. Waki, S. Kim, M. Kojima, and M. Muramatsu. Sums of Squares and Semidefinite Programming Relaxations for Polynomial Optimization Problems with Structured Sparsity. *SIAM Journal on Optimization*, 17(1):218–242, 2006.
[22] Hayato Waki, Sunyoung Kim, Masakazu Kojima, Masakazu Muramatsu, and Hiroshi Sugimoto. Algorithm 883: Sparsepop—a sparse semidefinite programming relaxation of polynomial optimization problems. *ACM Transactions on Mathematical Software (TOMS)*, 35(2):1–13, 2008.
[23] Jie Wang. ChordalGraph: A Julia Package to Handle Chordal Graphs. 2020.
[24] Jie Wang, Martina Maggio, and Victor Magron. SparseJSR: A Fast Algorithm to Compute Joint Spectral Radius via Sparse SOS Decompositions. *arXiv preprint arXiv:2008.11441*, 2020.
[25] Jie Wang and Victor Magron. Exploiting term sparsity in noncommutative polynomial optimization. *arXiv preprint arXiv:2010.06956*, 2020.
[26] Jie Wang, Victor Magron, and Jean-Bernard Lasserre. Chordal-TSSOS: a moment-SOS hierarchy that exploits term sparsity with chordal extension. *SIAM Journal on Optimization*, 2020. Accepted for publication.
[27] Jie Wang, Victor Magron, and Jean-Bernard Lasserre. TSSOS: A Moment-SOS hierarchy that exploits term sparsity. *SIAM Journal on Optimization*, 2020. Accepted for publication.
[28] Jie Wang, Victor Magron, Jean-Bernard Lasserre, and Ngoc Hoang Anh Mai. CS-TSSOS: Correlative and term sparsity for large-scale polynomial optimization. *arXiv:2005.02828*, 2020.
[29] Quan Zhou and Jakub Marecek. Proper learning of linear dynamical systems as a non-commutative polynomial optimisation problem. *arXiv preprint arXiv:2002.01444*, 2020.
[30] Quan Zhou, Jakub Marecek, and Robert N Shorten. Fairness in forecasting and learning linear dynamical systems. *arXiv preprint arXiv:2006.07315*, 2020.