SparseJSR: A Fast Algorithm to Compute Joint Spectral Radius via Sparse SOS Decompositions

Jie Wang, Martina Maggio and Victor Magron

Abstract— **This paper focuses on the computation of joint spectral radii (JSR), when the involved matrices are sparse. We provide a sparse variant of the procedure proposed by Parrilo and Jadbabaie in [24], to compute upper bounds of the JSR by means of sum-of-squares (SOS) relaxations. Our resulting iterative algorithm, called `SparseJSR`, is based on the *term sparsity* SOS (TSSOS) framework, developed by Wang, Magron and Lasserre in [37], yielding SOS decompositions of polynomials with arbitrary sparse support. `SparseJSR` exploits the sparsity of the input matrices to significantly reduce the computational burden associated with the JSR computation. Our algorithmic framework is then successfully applied to compute upper bounds for JSR, on randomly generated benchmarks as well as on problems arising from stability proofs of controllers, in relation with possible hardware and software faults.**

I. Introduction

Given a set of matrices $\mathcal{A} = \{A_1, \ldots, A_m\} \subseteq \mathbb{R}^{n\times n}$, the *joint spectral radius* (JSR) of $\mathcal{A}$ is defined by

$$\rho(\mathcal{A}) := \lim_{k\to\infty} \max_{\sigma\in\{1,\ldots,m\}^k} ||A_{\sigma_1}A_{\sigma_2}\cdots A_{\sigma_k}||^{\frac{1}{k}}, \quad (1)$$

which characterizes the maximal asymptotic growth rate of products of matrices from $\mathcal{A}$. Note that the value of $\rho(\mathcal{A})$ is independent of the choice of the norm used in (1). When $\mathcal{A}$ contains a single matrix, the JSR coincides with the usual spectral radius. Hence JSR can be viewed as a generalization of the usual spectral radius to the case of multiple matrices.

The concept of JSR was first introduced by Rota and Strang in [28] and since then has found applications in many areas such as the stability of switched linear dynamical systems, the continuity of wavelet functions, combinatorics and language theory, the capacity of some types of codes, the trackability of graphs. We refer the readers to [15] for a survey of the theory and applications of JSR.

Inspired by the various applications, there has been a lot of work on the computation of JSR; see e.g. [1], [4], [11], [12], [13], [24], [25] to name a few. Unfortunately, it turns out that the exact computation and even the approximation of JSR are notoriously difficult [31]. In fact, the problem of deciding whether $\rho(\mathcal{A}) \leq 1$ is undecidable even for $\mathcal{A}$ consisting of two matrices [5]. Therefore, various methods focus on computing lower bounds and upper bounds for JSR [1], [4], [11], [24].

Parrilo and Jadbabaie proposed in [24] a sum-of-squares (SOS) approach which uses semidefinite programming (SDP) to compute a sequence of upper bounds $\{\rho_{SOS,2d}(\mathcal{A})\}_{d\geq 1}$ for $\rho(\mathcal{A})$. They proved that the sequence $\{\rho_{SOS,2d}(\mathcal{A})\}_{d\geq 1}$ converges to $\rho(\mathcal{A})$ when d increases. In practice, mostly often even small d (e.g., $d = 1, 2$) can provide upper bounds of good quality for $\rho(\mathcal{A})$. Particularly, if the upper bound coincides with the lower bound, then we obtain the exact value of the JSR. However, the computational burden of the SOS approach grows rapidly when the matrix size or d increases. Given the current state of SDP solvers, this approach can only handle matrices of moderate sizes when $d \geq 2$.

For general polynomial optimization problems (POP), one way to reduce the computational cost of the associated SOS relaxations is to exploit the so-called *correlative sparsity pattern* arising from the variables of the POP [17], [34]. To build these sparse SOS relaxations, one relies on the *correlative sparsity pattern (csp) graph* of the POP. The nodes of the csp graph are the input variables and are connected via an edge when the corresponding variables appear in the same term of the objective function or in the same constraint involved in the POP. This approach was successfully used for several interesting applications, including rational functions [8], certified roundoff error bounds [21], [20], optimal powerflow problems [14], noncommutative problems [16], Lipschitz constant of ReLU networks [18], [9], robust geometric perception [40], positive definite forms [22] and polynomial matrix inequalities [41]. Recently, this methodology has been used to approximate sets defined by sparse input data [30], with the goal of tackling more control-oriented application, namely regions of attraction of sparse polynomial systems [29].

A complementary workaround is to take into account *term sparsity* (TS) of the input data to obtain sparse SOS representations, as recently studied in [36], [37], [38], yielding the so-called TSSOS framework. TSSOS relies on the *term sparsity pattern (tsp) graph*, related to the input polynomials. To build the associated sparse SOS relaxations, one connects the nodes of this graph (corresponding to monomials from a monomial basis), when the product of the corresponding monomials either appears in the supports of input polynomials or is an even degree monomial. Recent applications include learning and forecasting of linear systems [42], [43] via reformulation into noncommutative polynomial optimization and exploiting term sparsity to reduce the size of the associated relaxations. Note that term sparsity can be combined with correlative sparsity to reduce even further the size of the associated relaxations [23], [39].

The original underlying motivation of this paper was to apply term sparsity to improve the scalability of JSR computation arising from the study of deadline hit and deadline miss [19]. In this case, the computation of the control signal can fail due to a hardware and software fault, causing either no update or a delayed application of the control signal. The main application in this case is to determine how long the controller can operate in a faulty state (in which it does not complete the computation in due time, causing a deadline miss) before the stability of the system is compromised. The idea is to bound the JSR of products between state matrices associated to deadline hit and deadline miss by solving a POP [1]. For such JSR problems, matrices of large sizes issued from applications reveal certain kinds of sparsity in many cases. A natural question is: can we exploit the sparsity of matrices to improve the scalability of the SOS approach and to compute upper bounds more efficiently? In this paper, we address this specific question.

Contributions and outline: In Section II, we recall preliminary background about SOS polynomials, chordal graphs and approximation of JSR via SOS relaxations. To make the current paper as self-contained as possible, Section III is dedicated to detailed explanation about sparse SOS decompositions via generation of smaller monomial bases and exploitation of the block structure of Gram matrices. Our main contribution is described in Section IV, where we propose a so-called `SparseJSR` algorithm, based on the SOS approach, and in coordination with the sparsity of matrices appearing within the JSR computation. The performance of our `SparseJSR` algorithm is then illustrated in Section V, first on randomly generated benchmarks, then on benchmarks coming from the study of deadline hit/miss in [19]. We demonstrate its ability to compute upper bounds for JSR more efficiently than with the usual SOS approach. The algorithm is implemented in the open-source Julia library, also called `SparseJSR`, and is freely available.[1]. Although our sparse version of the SOS approach is a relaxation of the dense version and is not guaranteed to produce upper bounds for JSR as good as the dense one with the same relaxation order, the numerical experiments in this paper demonstrate that our sparse approach can indeed produce upper bounds of rather good quality but at a significantly cheaper cost compared to the dense approach.

II. Notation and Preliminaries

Let $\mathbf{x} = (x_1, \dots, x_n)$ be a tuple of variables and $\mathbb{R}[\mathbf{x}] = \mathbb{R}[x_1, \dots, x_n]$ be the ring of real n-variate polynomials. We use $\mathbb{R}[\mathbf{x}]_{2d}$ to denote the set of homogeneous polynomials of degree $2d$. A polynomial $f \in \mathbb{R}[\mathbf{x}]$ can be written as $f(\mathbf{x}) = \sum_{\boldsymbol{\alpha}\in\mathscr{A}} f_{\boldsymbol{\alpha}}\mathbf{x}^{\boldsymbol{\alpha}}$ with $f_{\boldsymbol{\alpha}} \in \mathbb{R}, \mathbf{x}^{\boldsymbol{\alpha}} = x_1^{\alpha_1}\cdots x_n^{\alpha_n}$. The *support* of f is defined by $\mathrm{supp}(f) := \{\boldsymbol{\alpha} \in \mathscr{A} \mid f_{\boldsymbol{\alpha}} \neq 0\}$. We use $|\cdot|$ for the cardinality of a set. For a nonempty finite set $\mathscr{A} \subseteq \mathbb{N}^n$, let $\mathbb{R}[\mathscr{A}]$ be the set of polynomials in $\mathbb{R}[\mathbf{x}]$ whose supports are contained in $\mathscr{A}$, i.e., $\mathbb{R}[\mathscr{A}] = \{f \in \mathbb{R}[\mathbf{x}] \mid \mathrm{supp}(f) \subseteq \mathscr{A}\}$ and let $\mathbf{x}^{\mathscr{A}}$ be the $|\mathscr{A}|$-dimensional column vector consisting of elements $\mathbf{x}^{\boldsymbol{\alpha}}, \boldsymbol{\alpha} \in \mathscr{A}$ (fix any ordering on $\mathbb{N}^n$). For convenience, we abuse notation in the sequel and note $\mathscr{B}$ (resp. $\boldsymbol{\beta}$) instead of $\mathbf{x}^{\mathscr{B}}$ (resp. $\mathbf{x}^{\boldsymbol{\beta}}$) a monomial set (resp. a monomial). For a positive integer r, let $\mathbf{S}^r$ be the set of $r \times r$ symmetric matrices and the set of $r \times r$ positive semidefinite (PSD) matrices is denoted by $\mathbf{S}_+^r$.

A. SOS polynomials

Given a homogeneous polynomial $f \in \mathbb{R}[\mathbf{x}]_{2d}$ with $d \in \mathbb{N}$, if there exist homogeneous polynomials $f_1, \dots, f_t \in \mathbb{R}[\mathbf{x}]_d$ such that $f = \sum_{i=1}^t f_i^2$, then we say that f is a *sum-of-squares* (SOS) polynomial. The set of SOS polynomials in $\mathbb{R}[\mathbf{x}]_{2d}$ is denoted by $\Sigma_{n,2d}$. For $d \in \mathbb{N}$, let $\mathbb{N}_d^n := \{\boldsymbol{\alpha} = (\alpha_i) \in \mathbb{N}^n \mid \sum_{i=1}^n \alpha_i = d\}$ and assume that $f \in \mathbb{R}[\mathbf{x}]_{2d}$. Then deciding whether $f \in \Sigma_{n,2d}$ is equivalent to verifying the existence of a PSD matrix Q (which is called a *Gram matrix* [26]) such that

$$f = (\mathbf{x}^{\mathbb{N}_d^n})^T Q \mathbf{x}^{\mathbb{N}_d^n}, \tag{2}$$

which can be formulated as a semidefinite program (SDP). The monomial basis $\mathbf{x}^{\mathbb{N}_d^n}$ used in (2) is called the *standard monomial basis*.

B. Chordal graphs and sparse matrices

An (undirected) *graph* $G(V, E)$ or simply G consists of a set of nodes V and a set of edges $E \subseteq \{\{v_i, v_j\} \mid (v_i, v_j) \in V \times V\}$. For a graph $G(V, E)$, a *cycle* of length k is a set of nodes $\{v_1, v_2, \dots, v_k\} \subseteq V$ with $\{v_k, v_1\} \in E$ and $\{v_i, v_{i+1}\} \in E$, for $i = 1, \dots, k-1$. A *chord* in a cycle $\{v_1, v_2, \dots, v_k\}$ is an edge $\{v_i, v_j\}$ that joins two nonconsecutive nodes in the cycle. A graph is called a *chordal graph* if all its cycles of length at least four have a chord. Chordal graphs include some common classes of graphs, such as complete graphs, line graphs and trees, and have applications in sparse matrix theory. Any non-chordal graph $G(V, E)$ can always be extended to a chordal graph $\overline{G}(V, \overline{E})$ by adding appropriate edges to E, which is called a *chordal-extension* of $G(V, E)$. A *clique* $C \subseteq V$ of G is a subset of nodes where $\{v_i, v_j\} \in E$ for any $v_i, v_j \in C$. If a clique C is not a subset of any other clique, then it is called a *maximal clique*. It is known that maximal cliques of a chordal graph can be enumerated efficiently in linear time in the number of nodes and edges of the graph [3].

Given a graph $G(V, E)$, a symmetric matrix Q with row and column indices labeled by V is said to have sparsity pattern G if $Q_{\boldsymbol{\beta}\boldsymbol{\gamma}} = Q_{\boldsymbol{\gamma}\boldsymbol{\beta}} = 0$ whenever $\boldsymbol{\beta} \neq \boldsymbol{\gamma}$ and $\{\boldsymbol{\beta}, \boldsymbol{\gamma}\} \notin E$. Let $\mathbf{S}_G$ be the set of symmetric matrices with sparsity pattern G. A matrix in $\mathbf{S}_G$ exhibits a *quasi block-diagonal* structure (after an appropriate permutation of rows and columns) as illustrated in Figure 1. Each block corresponds to a maximal clique of G. The maximal block size is the maximal size of maximal cliques of G, namely, the *clique number* of G. Note that there might be overlaps between blocks because different maximal cliques may share nodes.

Given a maximal clique C of $G(V, E)$, we define a matrix $P_C \in \mathbb{R}^{|C|\times|V|}$ as

$$[P_C]_{i\boldsymbol{\beta}} = \begin{cases} 1, & \text{if } C(i) = \boldsymbol{\beta}, \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

where $C(i)$ denotes the i-th node in C, sorted in the ordering compatibly with V. Note that $Q_C = P_C Q P_C^T \in \mathbf{S}^{|C|}$

[1] https://github.com/wangjie212/SparseJSR

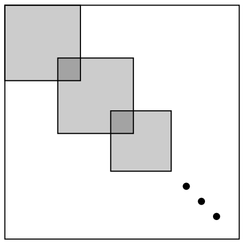

Fig. 1. The quasi block-diagonal structure of matrices in $\mathbf{S}_G$. The gray area indicates the positions of possible nonzero entries.

extracts a principal submatrix Q_C defined by the indices in the clique C from a symmetry matrix Q, and $Q = P_C^T Q_C P_C$ inflates a $|C| \times |C|$ matrix Q_C into a sparse $|V| \times |V|$ matrix Q.

PSD matrices with sparsity pattern G form a convex cone

$$\mathbf{S}_+^{|V|} \cap \mathbf{S}_G = \{Q \in \mathbf{S}_G \mid Q \succeq 0\}. \tag{4}$$

When the sparsity pattern graph G is chordal, the cone $\mathbf{S}_+^{|V|} \cap \mathbf{S}_G$ can be decomposed as a sum of simple convex cones, as stated in the following theorem.

Theorem 2.1 ([32], Theorem 9.2): Let $G(V, E)$ be a chordal graph and assume that $C_1, \ldots, C_t$ are all the maximal cliques of $G(V, E)$. Then a matrix $Q \in \mathbf{S}_+^{|V|} \cap \mathbf{S}_G$ if and only if there exist $Q_k \in \mathbf{S}_+^{|C_k|}$ for $k = 1, \ldots, t$ such that $Q = \sum_{k=1}^t P_{C_k}^T Q_k P_{C_k}$.

For more details about sparse matrices and chordal graphs, the reader may refer to [32].

C. Approximating joint spectral radii via SOS relaxations

The joint spectral radius (JSR) for a set of matrices $\mathcal{A} = \{A_1, \ldots, A_m\} \subseteq \mathbb{R}^{n \times n}$ is given by

$$\rho(\mathcal{A}) := \lim_{k \to \infty} \max_{\sigma \in \{1,\ldots,m\}^k} ||A_{\sigma_1} A_{\sigma_2} \cdots A_{\sigma_k}||^{\frac{1}{k}}. \tag{5}$$

Parrilo and Jadbabaie proposed to compute a sequence of upper bounds for $\rho(\mathcal{A})$ via SOS relaxations. The core idea is based on the following theorem.

Theorem 2.2 ([24], Theorem 2.2): Given a set of matrices $\mathcal{A} = \{A_1, \ldots, A_m\} \subseteq \mathbb{R}^{n \times n}$, let p be a strictly positive homogeneous polynomial of degree $2d$ that satisfies

$$p(A_i \mathbf{x}) \le \gamma^{2d} p(\mathbf{x}), \quad \forall \mathbf{x} \in \mathbb{R}^n, \quad i = 1, \ldots, m.$$

Then, $\rho(\mathcal{A}) \le \gamma$.

Replacing positive polynomials by more tractable SOS polynomials, Theorem 2.2 immediately suggests the following SOS relaxations indexed by $d \in \mathbb{N} \backslash \{0\}$ to compute a sequence of upper bounds for $\rho(\mathcal{A})$:

$$\rho_{\text{SOS},2d}(\mathcal{A}) := \inf_{p \in \mathbb{R}[\mathbf{x}]_{2d}, \gamma} \gamma \tag{6}$$
$$\text{s.t.} \begin{cases} p(\mathbf{x}) \in \Sigma_{n,2d}, \\ \gamma^{2d} p(\mathbf{x}) - p(A_i \mathbf{x}) \in \Sigma_{n,2d}, \ 1 \le i \le m. \end{cases}$$

The optimization problem (6) can be solved via SDP by bisection on γ. It was shown in [24] that the upper bound $\rho_{\text{SOS},2d}(\mathcal{A})$ satisfies the following theorem.

Theorem 2.3 ([24], Th 4.3): Let $\mathcal{A} = \{A_1, \ldots, A_m\} \subseteq \mathbb{R}^{n \times n}$. For any integer $d \ge 1$, $m^{-\frac{1}{2d}} \rho_{\text{SOS},2d}(\mathcal{A}) \le \rho(\mathcal{A}) \le \rho_{\text{SOS},2d}(\mathcal{A})$.

It is immediate from Theorem 2.3 that $\{\rho_{\text{SOS},2d}(\mathcal{A})\}_{d \ge 1}$ converges to $\rho(\mathcal{A})$ when d increases.

III. Sparse SOS decompositions

Deciding whether a polynomial f is SOS involves solving a SDP whose size scales combinatorially with the number of variables and the degree of f. When f is sparse, it is possible to exploit the sparsity to construct a SDP of smaller size and to reduce the computational burden. This includes two aspects: generating a smaller monomial basis and exploiting block structure for Gram matrices.

A. Generating a smaller monomial basis

Given a polynomial $f \in \mathbb{R}[\mathbf{x}]$, the *Newton polytope* of f is the convex hull of the support of f. It is known that the monomial basis $\mathbb{N}_d^n$ used in (2) can be replaced by the integer points in half of the Newton polytope of f, i.e., by

$$\mathscr{B} = \frac{1}{2}\text{New}(f) \cap \mathbb{N}^n \subseteq \mathbb{N}_d^n. \tag{7}$$

See, e.g., [27] for a proof.

The following so-called `GenerateBasis` algorithm to generate a smaller monomial basis for (2) was proposed in [38].

Algorithm 1 `GenerateBasis`

Input: $\mathscr{A} = \text{supp}(f) \subseteq \mathbb{N}_{2d}^n$ and the initial monomial basis $\mathscr{B} = \mathbb{N}_d^n$

Output: An increasing chain of potential monomial bases $(\mathscr{B}_p)_{p \ge 1}$

Set $\mathscr{B}_0 := \emptyset$ and $p = 0$;
while $p = 0$ or $\mathscr{B}_p \ne \mathscr{B}_{p-1}$ **do**
  $p := p + 1$;
  Set $\mathscr{B}_p := \emptyset$;
  for each pair $\{\boldsymbol{\beta}, \boldsymbol{\gamma}\}$ of $\mathscr{B}$ **do**
    if $\boldsymbol{\beta} + \boldsymbol{\gamma} \in \mathscr{A} \cup (2\mathscr{B}_{p-1})$ **then**
      $\mathscr{B}_p := \mathscr{B}_p \cup \{\boldsymbol{\beta}, \boldsymbol{\gamma}\}$;
    end if
  end for
end while
return $(\mathscr{B}_p)_{p \ge 1}$;

The output of `GenerateBasis` is an increasing chain of monomial sets:

$$\mathscr{B}_1 \subseteq \mathscr{B}_2 \subseteq \mathscr{B}_3 \subseteq \cdots \subseteq \mathbb{N}_d^n.$$

Each $\mathscr{B}_p$ can serve as a candidate monomial basis. In practice, if indexing the unknown Gram matrix from (2) by $\mathscr{B}_p$ leads to an infeasible SDP, then we turn to $\mathscr{B}_{p+1}$ until a feasible SDP is retrieved. In many cases, the algorithm `GenerateBasis` can provide a monomial basis smaller than the one given by (7); see [38] for such examples.

Remark 3.1: For all tested examples, it turns out that $\mathscr{B}_1$ is always qualified to serve as a monomial basis.

B. Term sparsity patterns

To derive a block structure for Gram matrices, we recall the concept of term sparsity patterns [36], [37], [38], [39].

Definition 3.2: Let $f(\mathbf{x}) \in \mathbb{R}[\mathbf{x}]$ with $\mathrm{supp}(f) = \mathscr{A}$. Assume that $\mathscr{B}$ is a monomial basis. The *term sparsity pattern graph* $G(V, E)$ of f is defined by $V = \mathscr{B}$ and

$$E = \{\{\boldsymbol{\beta}, \boldsymbol{\gamma}\} \mid \boldsymbol{\beta}, \boldsymbol{\gamma} \in V, \boldsymbol{\beta} \neq \boldsymbol{\gamma}, \boldsymbol{\beta} + \boldsymbol{\gamma} \in \mathscr{A} \cup 2\mathscr{B}\}, \quad (8)$$

where $2\mathscr{B} = \{2\boldsymbol{\beta} \mid \boldsymbol{\beta} \in \mathscr{B}\}$.
For a term sparsity pattern graph $G(V, E)$, we denote a chordal-extension of G by $\overline{G}(V, \overline{E})$.

Remark 3.3: For a graph G, the chordal-extension of G is usually not unique. We prefer a chordal-extension with the least clique number. Finding a chordal-extension with the least clique number is an NP-complete problem in general. Fortunately, several heuristic algorithms are known to efficiently produce a good approximation [6].

Example 3.4: Consider the polynomial $f = x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 + x_6^2 + x_1x_2 + x_2x_3 + x_3x_4 + x_4x_5 + x_5x_6 + x_6x_1$. See Figure 2 for the term sparsity pattern graph G of f and a chordal-extension $\overline{G}$ of G.

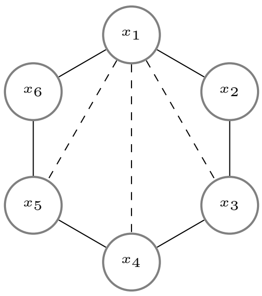

Fig. 2. Term sparsity pattern graph and chordal-extension for Example 3.4. The dashed edges are added in the process of chordal-extension.

Given a sparse SOS polynomial $f(\mathbf{x}) \in \mathbb{R}[\mathscr{A}]$ and a monomial basis $\mathscr{B}$, generally a Gram matrix for f is not necessarily sparse. Let G be the term sparsity pattern graph of f and $\overline{G}$ a chordal-extension. To get a sparse SOS decomposition of f, we then impose the sparsity pattern $\overline{G}$ to the Gram matrix for f, i.e., we consider the following subset of SOS polynomials in $\Sigma_{n,2d}$:

$$\Sigma_{\mathscr{A}} := \{f \in \mathbb{R}[\mathscr{A}] \mid \exists Q \in \mathbf{S}_+^{|\mathscr{B}|} \cap \mathbf{S}_{\overline{G}} \text{ s.t. } f = (\mathbf{x}^{\mathscr{B}})^T Q \mathbf{x}^{\mathscr{B}}\}.$$

Theorem 2.1 enables us to give the following sparse SOS decompositions for polynomials in $\Sigma_{\mathscr{A}}$.

Theorem 3.5 ([36], Theorem 3.3): Given $\mathscr{A} \subseteq \mathbb{N}^n$, assume that $\mathscr{B} = \{\boldsymbol{\omega}_1, \ldots, \boldsymbol{\omega}_r\}$ is a monomial basis and G is the term sparsity pattern graph. Let $C_1, C_2, \ldots, C_t \subseteq V$ denote the maximal cliques of $\overline{G}$ and $\mathscr{B}_k = \{\boldsymbol{\omega}_i \in \mathscr{B} \mid i \in C_k\}, k = 1, 2, \ldots, t$. Then, $f(\mathbf{x}) \in \Sigma_{\mathscr{A}}$ if and only if there exist $f_k(\mathbf{x}) = (\mathbf{x}^{\mathscr{B}_k})^T Q_k \mathbf{x}^{\mathscr{B}_k}$ with $Q_k \in \mathbf{S}_+^{|C_k|}$ for $k = 1, \ldots, t$ such that

$$f(\mathbf{x}) = \sum_{k=1}^{t} f_k(\mathbf{x}). \quad (9)$$

By virtue of Theorem 3.5, checking membership in $\Sigma_{\mathscr{A}}$ boils down to solving an SDP problem involving SDP matrices of small sizes if each maximal clique of $\overline{G}$ has a small size with respect to the original matrix. This might significantly reduce the overall computational cost.

IV. The SparseJSR Algorithm

In this section, we propose an algorithm for computing JSR based on the sparse SOS decompositions discussed in the previous section. Let $\mathcal{A} = \{A_1, \ldots, A_m\} \subseteq \mathbb{R}^{n \times n}$. First we derive a sparse support for $p(\mathbf{x})$ in (6). Fixing a relaxation order d, let $p_0(\mathbf{x}) = x_1^{2d} + x_2^{2d} + \ldots + x_n^{2d}$ and $\mathscr{A}_0 = \mathrm{supp}(p_0)$. Then define

$$\mathscr{A} := \mathscr{A}_0 \cup \bigcup_{i=1}^{m} \mathrm{supp}(p_0(A_i\mathbf{x})). \quad (10)$$

By restricting $p(\mathbf{x})$ to polynomials with the sparse support $\mathscr{A}$, (6) now reads as

$$\rho_{\mathrm{SSOS},2d}(\mathcal{A}) := \inf_{p \in \mathbb{R}[\mathscr{A}], \gamma} \gamma \quad (11)$$
$$\text{s.t.} \begin{cases} p(\mathbf{x}) \in \Sigma_{n,2d}, \\ \gamma^{2d} p(\mathbf{x}) - p(A_i\mathbf{x}) \in \Sigma_{n,2d}, \ 1 \le i \le m. \end{cases}$$

Let $\mathscr{A}_i = \mathscr{A} \cup \mathrm{supp}(p(A_i\mathbf{x})$ for $i = 1, \ldots, m$. To exploit the sparsity in (11), we further replace $\Sigma_{n,2d}$ with $\Sigma_{\mathscr{A}}$ or $\Sigma_{\mathscr{A}_i}$ in (11). Consequently we obtain

$$\rho_{\mathrm{TSSOS},2d}(\mathcal{A}) := \inf_{p \in \mathbb{R}[\mathscr{A}], \gamma} \gamma \quad (12)$$
$$\text{s.t.} \begin{cases} p(\mathbf{x}) \in \Sigma_{\mathscr{A}}, \\ \gamma^{2d} p(\mathbf{x}) - p(A_i\mathbf{x}) \in \Sigma_{\mathscr{A}_i}, \ 1 \le i \le m. \end{cases}$$

As in the dense case, the problem (12) can be solved via SDP by bisection on γ. Moreover, we have

Theorem 4.1: Let $\mathcal{A} = \{A_1, \ldots, A_m\} \subseteq \mathbb{R}^{n \times n}$. For any integer $d \geq 1$, one has $\rho_{\mathrm{SOS},2d}(\mathcal{A}) \leq \rho_{\mathrm{SSOS},2d}(\mathcal{A}) \leq \rho_{\mathrm{TSSOS},2d}(\mathcal{A})$.

Proof: For any fixing $d \in \mathbb{N}\backslash\{0\}$, it is clear that the feasible set of (12) is contained in the feasible set of (11), which is contained in the feasible set of (6). This yields the desired conclusion. ■

Overall, we design the algorithm `SparseJSR` to compute a sequence of upper bounds for the JSR of a set of matrices. The correctness of `SparseJSR` is guaranteed by Theorem 2.2 and Theorem 4.1.

V. Numerical Experiments

In this section, we present numerical experiments for the proposed algorithm `SparseJSR`, implemented in a tool also named `SparseJSR`, which is written as a Julia package, and based on the `TSSOS` package used in [37], [38], [39]. `SparseJSR` utilizes the Julia packages LightGraphs [7] to handle graphs, ChordalGraph [35] to generate a chordal-extension and JuMP [10] to model SDP. Finally, `SparseJSR` relies on the SDP solver MOSEK [2] to solve SDP. For the comparison purpose, we also implement the dense SOS relaxation (6) in `SparseJSR` using the same SDP solver MOSEK. For all examples, we set the tolerance $\epsilon = 1 \times 10^{-5}$ for bisection. To measure the quality of upper bounds that we obtain, we compute a lower bound for the JSR using the Matlab JSR toolbox [33]. All examples were

Algorithm 2 `SparseJSR`

Input: A set of matrices $\mathcal{A} = \{A_1, A_2, \ldots, A_m\} \subseteq \mathbb{R}^{n\times n}$, an initial lower bound lb for $\rho(\mathcal{A})$, an initial upper bound ub for $\rho(\mathcal{A})$, a relaxation order d and a tolerance ϵ for the bisection

Output: An upper bound $\rho_{\mathrm{TSSOS},2d}(\mathcal{A})$ for $\rho(\mathcal{A})$

```
Let 𝒜 be as in (10);
Compute monomial bases for (12) with the GenerateBasis algorithm;
Let ξ = lb and η = ub;
while |ξ − η| > ϵ do
    Let γ = (ξ+η)/2;
    if (12) is feasible then η = γ;
    else ξ = γ;
    end if
end while
return η;
```

computed on an Intel Core i5-8265U@1.60GHz CPU with 8GB RAM memory. The notations that we use are as follows: m (resp. n) stands for the number (resp. size) of matrices, lb (resp. ub) stands for the lower (resp. upper) bound for the JSR, d is the SOS relaxation order, mb stands for the maximal size of blocks and "time" refers to the running time in seconds. We consider randomly generated examples and examples arising from the study of deadline hit/miss in [19].

A. Randomly generated examples

We generate random matrices as follows: first call the function "erdos_renyi" in the Julia packages LightGraphs to generate a random (directed) graph G with n nodes and $n+10$ edges; for each edge (i, j) of G, put a random number in $[-1, 1]$ on the position (i, j) of the matrix and put zeros for other positions. We compute the JSR for pairs of such matrices using the first-order SOS relaxations. The results are shown in Table I. One can see that the sparse relaxation is much more efficient than the dense relaxation. The dense relaxation takes over 3600 s when the size of matrices is more than 100 while our sparse relaxation can easily handle matrices of size 120 within 12 s. Both the dense relaxation and the sparse relaxation produce upper bounds which are within 0.05 larger than the corresponding lower bounds.

B. Examples from control systems

Here we consider examples from [19], where the dynamics of closed-loop systems are given by the combination of a plant and a one-step delay controller that stabilizes the plant. The closed-loop system evolves according to a either a completed or a missed computation. In the case of a deadline hit, the closed-loop state matrix is A_H. In the case of a deadline miss, the associated closed-loop state matrix is A_M. The computational platform (hardware and software) ensures that no more than $m-1$ deadlines are missed consecutively. The set of possible realisations $\mathcal{A}$ of such a system contains either a single hit or at most $m-1$ misses followed by a hit,

TABLE I

RANDOMLY GENERATED EXAMPLES WITH $d = 1$ AND $m = 2$

n	lb	Sparse ($d=1$)			Dense ($d=1$)		
		time	ub	mb	time	ub	mb
20	0.7894	0.74	0.8192	10	1.88	0.7967	20
30	0.8502	1.65	0.8666	10	7.79	0.8523	30
40	0.9446	2.68	0.9446	14	25.6	0.9446	40
50	0.8838	2.97	0.9102	14	55.9	0.8838	50
60	0.7612	3.64	0.7843	13	171	0.7612	60
70	0.9629	4.35	0.9629	11	308	0.9629	70
80	0.9345	5.95	0.9399	15	743	0.9345	80
90	0.8020	6.27	0.8465	14	1282	0.8020	90
100	0.8642	8.15	0.9132	13	2568	0.8659	100
110	0.8355	9.59	0.8839	15	-	-	-
120	0.7483	11.7	0.7735	16	-	-	-

namely $\mathcal{A} := \{A_H A_M^i \mid 0 \leq i \leq m-1\}$. Then, the closed-loop system that can switch between the realisations included in $\mathcal{A}$ is asymptotically stable if and only if $\rho(\mathcal{A}) < 1$. This gives an indication for scheduling and control co-design, in which the hardware and software platform must guarantee that the maximum number of deadlines missed consecutively does not interfere with stability requirements.

In Table II and Table III, we report the results obtained for various control systems with n states, under $m-1$ deadline misses, by applying the dense and sparse relaxations with order $d = 1$ and $d = 2$, respectively. The examples are randomly generated, i.e., our script generates a random system and then tries to control it.

In Table II, we fix $m = 5$ and vary n from 20 to 110. For these examples, surprisingly the dense and sparse relaxations with relaxation order $d = 1$ always produce the same upper bounds. As we can see from the table, the sparse relaxation is more scalable and more efficient than the dense one.

In Table III, we vary m from 2 to 11 and vary n from 6 to 24. For each instance, one has $mb = 10$ for the sparse relaxation. The column "ub" indicates the upper bound given by the SOS relaxation of order $d = 1$. For these examples, with relaxation order $d = 2$, the sparse relaxation produces upper bounds that are very close to those given by the dense relaxation. And again the sparse relaxation is more scalable and more efficient than the dense one.

REFERENCES

[1] A. A. AHMADI, R. M. JUNGERS, P. A. PARRILO, AND M. ROOZBEHANI, *Joint spectral radius and path-complete graph lyapunov functions*, SIAM Journal on Control and Optimization, 52 (2014).

[2] M. APS, *The MOSEK optimization toolbox. Version 8.1.*, 2017.

[3] J. R. BLAIR AND B. PEYTON, *An introduction to chordal graphs and clique trees*, in Graph theory and sparse matrix computation, Springer, 1993, pp. 1–29.

[4] V. D. BLONDEL AND Y. NESTEROV, *Polynomial-time computation of the joint spectral radius for some sets of nonnegative matrices*, SIAM Journal on Matrix Analysis and Applications, 31 (2010), pp. 865–876.

[5] V. D. BLONDEL AND J. N. TSITSIKLIS, *The boundedness of all products of a pair of matrices is undecidable*, Systems & Control Letters, 41 (2000), pp. 135–140.

[6] H. L. BODLAENDER AND A. M. KOSTER, *Treewidth computations i. upper bounds*, Information and Computation, 208 (2010).

[7] S. BROMBERGER, J. FAIRBANKS, AND OTHER CONTRIBUTORS, *Juliagraphs/lightgraphs.jl: an optimized graphs package for the julia programming language*, 2017.

TABLE II

RESULTS FOR CONTROL SYSTEMS WITH $d = 1$ AND $m = 5$

n	lb	Sparse ($d = 1$)			Dense ($d = 1$)		
		time	ub	mb	time	ub	mb
20	0.9058	1.78	0.9316	12	9.92	0.9316	20
20	0.8142	1.62	0.8142	12	9.08	0.8142	20
30	1.4682	4.30	1.5132	14	57.8	1.5131	30
30	1.0924	4.42	1.0961	14	65.4	1.0961	30
40	1.1648	9.29	1.1977	16	249	1.1977	30
40	0.9772	9.69	0.9804	16	259	0.9804	30
50	1.3153	17.3	1.3248	18	660	1.3248	50
50	1.1884	17.5	1.1884	18	680	1.1884	50
60	1.8366	29.7	1.8820	20	2049	1.8820	60
60	1.3259	30.7	1.3259	20	1776	1.3259	60
70	1.8135	54.2	1.8578	22	-	-	-
70	1.2727	53.9	1.2727	22	-	-	-
80	2.3005	85.3	2.3445	24	-	-	-
80	1.4262	85.6	1.4262	24	-	-	-
90	1.8745	133	1.9020	26	-	-	-
90	1.4452	132	1.4452	26	-	-	-
100	2.2316	196	2.2733	28	$*$	$*$	$*$
100	1.5267	195	1.5267	28	$*$	$*$	$*$
110	2.3597	280	2.3943	30	$*$	$*$	$*$
110	1.5753	287	1.5753	30	$*$	$*$	$*$

TABLE III

RESULTS FOR CONTROL SYSTEMS WITH $d = 2$

m	n	lb	ub	Sparse ($d = 2$)		Dense ($d = 2$)		
				time	ub	time	ub	mb
2	6	0.9464	0.9782	0.42	0.9547	1.87	0.9539	21
3	8	0.7218	0.7467	0.60	0.7310	13.4	0.7305	36
4	10	0.7458	0.7738	0.75	0.7564	107	0.7554	55
5	12	0.8601	0.8937	1.08	0.8706	1157	0.8699	78
6	14	0.7875	0.8107	1.32	0.7958	-	-	-
7	16	1.1110	1.1531	1.81	1.1182	$*$	$*$	$*$
8	18	1.0487	1.0881	2.05	1.0569	$*$	$*$	$*$
9	20	0.7570	0.7808	2.52	0.7660	$*$	$*$	$*$
10	22	0.9911	1.0315	2.70	1.0002	$*$	$*$	$*$
11	24	0.7339	0.7530	3.67	0.7418	$*$	$*$	$*$

[8] F. BUGARIN, D. HENRION, AND J. B. LASSERRE, *Minimizing the sum of many rational functions*, Mathematical Programming Computation, 8 (2016), pp. 83–111.

[9] T. CHEN, J.-B. LASSERRE, V. MAGRON, AND E. PAUWELS, *Semialgebraic Optimization for Bounding Lipschitz Constants of ReLU Networks*, arXiv:2002.03657, (2020).

[10] I. DUNNING, J. HUCHETTE, AND M. LUBIN, *JuMP: A modeling language for mathematical optimization*, SIAM Review, 59 (2017), pp. 295–320.

[11] G. GRIPENBERG, *Computing the joint spectral radius*, Linear Algebra and its Applications, 234 (1996), pp. 43–60.

[12] N. GUGLIELMI AND V. PROTASOV, *Exact computation of joint spectral characteristics of linear operators*, Foundations of Computational Mathematics, 13 (2013), pp. 37–97.

[13] N. GUGLIELMI AND M. ZENNARO, *Finding extremal complex polytope norms for families of real matrices*, SIAM Journal on Matrix Analysis and Applications, 31 (2009), pp. 602–620.

[14] C. JOSZ AND D. K. MOLZAHN, *Lasserre hierarchy for large scale polynomial optimization in real and complex variables*, SIAM Journal on Optimization, 28 (2018), pp. 1017–1048.

[15] R. JUNGERS, *The joint spectral radius: theory and applications*, vol. 385, Springer Science & Business Media, 2009.

[16] I. KLEP, V. MAGRON, AND J. POVH, *Sparse noncommutative polynomial optimization*, arXiv:1909.00569, (2019).

[17] J.-B. LASSERRE, *Convergent SDP-relaxations in polynomial optimization with sparsity*, SIAM Journal on Optimization, 17 (2006), pp. 822–843.

[18] F. LATORRE, P. ROLLAND, AND V. CEVHER, *Lipschitz constant estimation of neural networks via sparse polynomial optimization*, arXiv:2004.08688, (2020).

[19] M. MAGGIO, A. HAMANN, E. MAYER-JOHN, AND D. ZIEGENBEIN, *Control-system stability under consecutive deadline misses constraints*, in 32nd Euromicro Conference on Real-Time Systems (ECRTS 2020), Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2020.

[20] V. MAGRON, *Interval enclosures of upper bounds of roundoff errors using semidefinite programming*, ACM Transactions on Mathematical Software (TOMS), 44 (2018), pp. 1–18.

[21] V. MAGRON, G. CONSTANTINIDES, AND A. DONALDSON, *Certified Roundoff Error Bounds Using Semidefinite Programming*, ACM Trans. Math. Softw., 43 (2017), pp. 1–34.

[22] N. H. A. MAI, V. MAGRON, AND J.-B. LASSERRE, *A sparse version of reznick's positivstellensatz*, arXiv:2002.05101, (2020).

[23] J. MILLER, Y. ZHENG, M. SZNAIER, AND A. PAPACHRISTODOULOU, *Decomposed structured subsets for semidefinite and sum-of-squares optimization*, arXiv:1911.12859, (2019).

[24] P. A. PARRILO AND A. JADBABAIE, *Approximation of the joint spectral radius using sum of squares*, Linear Algebra and its Applications, 428 (2008), pp. 2385–2402.

[25] V. Y. PROTASOV, R. M. JUNGERS, AND V. D. BLONDEL, *Joint spectral characteristics of matrices: a conic programming approach*, SIAM Journal on Matrix Analysis and Applications, 31 (2010), pp. 2146–2162.

[26] B. REZNICK ET AL., *Extremal psd forms with few terms*, Duke mathematical journal, 45 (1978), pp. 363–374.

[27] ———, *Extremal psd forms with few terms*, Duke mathematical journal, 45 (1978), pp. 363–374.

[28] G.-C. ROTA AND W. STRANG, *A note on the joint spectral radius*, (1960).

[29] M. TACCHI, C. CARDOZO, D. HENRION, AND J. LASSERRE, *Approximating regions of attraction of a sparse polynomial differential system*, arXiv:1911.09500, (2019).

[30] M. TACCHI, T. WEISSER, J.-B. LASSERRE, AND D. HENRION, *Exploiting Sparsity for Semi-Algebraic Set Volume Computation*, preprint arXiv:1902.02976, (2019).

[31] J. N. TSITSIKLIS AND V. D. BLONDEL, *The Lyapunov exponent and joint spectral radius of pairs of matrices are hardwhen not impossibleto compute and to approximate*, Mathematics of Control, Signals and Systems, 10 (1997), pp. 31–40.

[32] L. VANDENBERGHE, M. S. ANDERSEN, ET AL., *Chordal graphs and semidefinite optimization*, Foundations and Trends® in Optimization, 1 (2015), pp. 241–433.

[33] G. VANKEERBERGHEN, J. HENDRICKX, AND R. M. JUNGERS, *Jsr: A toolbox to compute the joint spectral radius*, in Proceedings of the 17th international conference on Hybrid systems: computation and control, 2014, pp. 151–156.

[34] 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 (2006), pp. 218–242.

[35] J. WANG, *ChordalGraph: A Julia Package to Handle Chordal Graphs*, (2020).

[36] J. WANG, H. LI, AND B. XIA, *A new sparse SOS decomposition algorithm based on term sparsity*, in Proceedings of the 2019 on International Symposium on Symbolic and Algebraic Computation, 2019, pp. 347–354.

[37] J. WANG, V. MAGRON, AND J.-B. LASSERRE, *TSSOS: A moment-SOS hierarchy that exploits term sparsity*, arXiv:1912.08899, (2019).

[38] ———, *Chordal-TSSOS: a moment-SOS hierarchy that exploits term sparsity with chordal extension*, arXiv:2003.03210, (2020).

[39] J. WANG, V. MAGRON, J.-B. LASSERRE, AND N. H. A. MAI, *CS-TSSOS: Correlative and term sparsity for large-scale polynomial optimization*, arXiv:2005.02828, (2020).

[40] H. YANG AND L. CARLONE, *One ring to rule them all: Certifiably robust geometric perception with outliers*, arXiv:2006.06769, (2020).

[41] Y. ZHENG AND G. FANTUZZI, *Sum-of-squares chordal decomposition of polynomial matrix inequalities*, arXiv:2007.11410, (2020).

[42] Q. ZHOU AND J. MARECEK, *Proper learning of linear dynamical systems as a non-commutative polynomial optimisation problem*, arXiv:2002.01444, (2020).

[43] Q. ZHOU, J. MARECEK, AND R. N. SHORTEN, *Fairness in forecasting and learning linear dynamical systems*, arXiv:2006.07315, (2020).