Mathematical Programming Computation (2024) 16:503–534
https://doi.org/10.1007/s12532-024-00264-w

FULL LENGTH PAPER

Solving clustered low-rank semidefinite programs arising from polynomial optimization

Nando Leijenhorst[1] · **David de Laat**[1]

Received: 31 March 2023 / Accepted: 21 August 2024 / Published online: 6 September 2024

Abstract
We study a primal-dual interior point method specialized to clustered low-rank semidefinite programs requiring high precision numerics, which arise from certain multivariate polynomial (matrix) programs through sums-of-squares characterizations and sampling. We consider the interplay of sampling and symmetry reduction as well as a greedy method to obtain numerically good bases and sample points. We apply this to the computation of three-point bounds for the kissing number problem, for which we show a significant speedup. This allows for the computation of improved kissing number bounds in dimensions 11 through 23. The approach performs well for problems with bad numerical conditioning, which we show through new computations for the binary sphere packing problem.

Keywords Semidefinite programming · Primal-dual interior point method · Low-rank constraints · Symmetry reduction · Packing problems · Sum-of-squares polynomials

Mathematics Subject Classification 90C22 · 90C23 · 52C17 · 90-04

1 Introduction

In discrete geometry, many of the best-known bounds on the optimal cardinality of spherical codes [1–5], optimal sphere packing densities [6, 7], optimal densities for packings with nonspherical shapes [8], and optimal ground state energies [9–11] are obtained using linear programming and semidefinite programming bounds. Similar approaches are used in analytic number theory [12] and the conformal bootstrap [13].

These bounds are derived using constraints on k-point correlations for small k, which leads to conic optimization problems involving positive semidefinite matrix variables as well as polynomial inequality constraints in a modest number of variables.

✉ Nando Leijenhorst
n.m.leijenhorst@tudelft.nl

David de Laat
d.delaat@tudelft.nl

1 Delft University of Technology, Mekelweg 4, 2628 CD Delft, The Netherlands

Such problems are then solved using semidefinite programming via sums-of-squares characterizations for the polynomial constraints.

For applications in discrete geometry, we often need solutions of rather high precision, so we need to use *second-order* interior point methods, which have linear convergence. Moreover, due to the orthogonal bases in the formulations of the problems, the problems have seemingly unavoidable bad numerical conditioning. In practice, computations are therefore performed using the general purpose semidefinite programming solvers `SDPA-QD` and `SDPA-GMP` [14], which both use high-precision numerics, and computations regularly take weeks to complete (see, e.g., [5]).

When the constraint matrices defining a semidefinite program are of low rank, this can be exploited in the interior-point method [15, 16], which has been used in the solvers [13, 17, 18]. As observed by Parrilo and Löfberg [15] rank 1 constraints naturally appear from sum-of-squares characterizations when sampling is used as opposed to coefficient matching.

In [13, 19] Simmons-Duffin develops a high-precision solver that uses this rank 1 structure. The solver exploits clustering in the constraints, supports parallelization, and works very well in practice. The implementation however has been designed for problems in the conformal bootstrap and only supports a specific form of semidefinite programs that arise from univariate polynomial optimization problems. Inspired by this success, the goal of this paper is to explore the use of low-rank constraints for solving optimization problems such as those arising in discrete geometry.

That low-rank constraints can be exploited is not obvious, since problems in discrete geometry often have large symmetry groups, and symmetry reduction leads to constraints on the positive semidefinite matrix variables which may be of large rank and which are no longer sparse due to the sampling approach. Moreover, it also leads to multivariate polynomial inequality constraints which are invariant under certain group actions, and exploiting this leads to constraints matrices of rank greater than one.

Our first contribution is the implementation of a high-precision, primal-dual, interior-point solver that can exploit more general low-rank structures for the constraint matrices than only rank 1 constraints.[1] The solver is written in the high-level language Julia [20], and is implemented in such a way that fast matrix-matrix multiplication can be exploited (using Arb [21]). It comes with a user-friendly interface for modeling problems involving both low-rank semidefinite constraints, as well as low-rank polynomial constraints, and it can automatically convert between these. Similar to [13] the solver can exploit clustering of the constraints (where clusters of positive semidefinite matrix blocks are linked through free variables), and the solver has a custom parallelization approach tailored to problems where we have fewer clusters, but many samples per cluster. Note that many of the features of our implementation are already present in an existing solver, but none of the existing solvers implements all of them at the same time.

Secondly, we study the interplay of sampling and symmetry reduction, which has not been done before. We give necessary and sufficient conditions on the sample set in the presence of symmetry. We furthermore show empirically that a greedy approach

[1] See github.com/nanleij/ClusteredLowRankSolver.jl for source code and documentation.

to finding good samples, and transforming the bases to be orthogonal with respect to these samples, works well in this setting. We also investigate an approach where we remove dense constraint matrices by parameterizing them by free variables, which allows for more clustering; see Sect. 6.1.

Our third contribution is to emperically show the speed and stability of the approach described in the previous paragraphs by considering two applications from discrete geometry. First, we consider the three-point bound for the kissing number problem [3]. We consider this problem because it involves invariant, multivariate polynomial inequality constraints and because it has additional positive semidefinite matrix variables for which the clustering approach we use plays a role. Moreover, it is an important problem in discrete geometry for which extensive computations have already been performed. We show a significant speedup compared to previous computations performed with `SDPA-GMP`, by a factor 28 for the most extensive computations previously performed. This allows us to perform computations using polynomials of degree 40 as opposed to degree 32 (for which extrapolation shows this approach would have been faster by a factor 40), which also results in improved kissing number bounds in dimension 11-23.

Then we consider the adaptation of the Cohn-Elkies bound for the binary sphere packing problem [7]. We show this bound can be written using matrix polynomial inequality constraints, and we use this to perform new computations which highlight the numerical stability of the solver. These computations show the bounds are not necessarily convex, can beat Florian's bound in dimensions 2, and may converge to the optimal density in dimensions 8 and 24 as the ratio of the radii goes to 0.

These two representative examples show that exploiting low-rank constraints can be very beneficial for the k-point bounds arising in discrete geometry. We expect this will not just speed up existing computations, but will also allow for tackling more difficult problems which were previously out of reach.

2 A specialized interior point method

In this section we give an exposition of the primal-dual interior-point method for semidefinite programming as used by `SDPB` [13], which in turn builds on `SDPA` [14]. We generalize the method to a very general low-rank structure (see (2) and (3)), and we show how this can be exploited in the computation of the Schur complement matrix in a way that fast matrix-matrix multiplication can be employed (which is especially beneficial because we use high-precision arithmetic). Because our applications consist of problems in extremal geometry, which typically have few clusters and a large number of constraints within a cluster, our parallelization strategy is different from `SDPB`. The interior point method uses the XZ search direction [22–24], the predictor-corrector step due to Mehrotra [25], and the Lanczos algorithm for computing step lengths [26].

When translating polynomial constraints into semidefinite constraints (see Sect. 3), one obtains for each polynomial constraint a number of semidefinite constraints which use the same positive semidefinite matrix variables. By using sampling it is possible to keep the rank of the constraint matrices low [15]. Together, this leads to a clustered low-rank semidefinite program, with clusters of constraints using the same positive semidefinite variables, and low-rank constraint matrices. We assume these clusters are connected only through free scalar variables.

We therefore consider semidefinite programs with J clusters of the form

$$\begin{aligned} &\text{maximize} && \sum_{j=1}^{J}\langle C^j, Y^j\rangle + \langle c, y\rangle && \\ &\text{subject to} && \left\langle A_*^j, Y^j\right\rangle + B^j y = b^j, && j = 1, \dots, J \\ & && Y^j \succeq 0, && j = 1, \dots, J, \end{aligned} \tag{1}$$

where we optimize over the vector of free variables y and the positive semidefinite block matrices $Y^j = \mathrm{diag}(Y^{j,1}, \dots, Y^{j,L_j})$. Here $\langle c, y\rangle$ is the Euclidean inner product, and we use the notation

$$\langle A_*^j, Y^j\rangle = \left(\langle A_t^j, Y^j\rangle\right)_{t\in T_j},$$

where $\langle A_t^j, Y^j\rangle$ is the trace inner product.

The semidefinite program is defined by the symmetric matrices C^j and A_t^j, the matrices B^j, and the vectors $c \in \mathbb{R}^N$ and $b^j \in \mathbb{R}^{T_j}$. We assume the matrix A_t^j is of the form

$$A_t^j = \bigoplus_{l=1}^{L_j} \sum_{r,s=1}^{R_j(l)} A_t^{j,l}(r, s) \otimes E_{r,s}^{R_j(l)}, \tag{2}$$

with $A_t^{j,l}(r, s)$ a matrix of low rank and $A_t^{j,l}(r, s)^{\mathsf{T}} = A_t^{j,l}(s, r)$. Here $E_{r,s}^n$ is the $n \times n$ matrix with a one at position (r, s) and zeros otherwise.

Internally, we represent the blocks $A_t^{j,l}(r, s)$ in the form

$$\sum_i \lambda_i v_i w_i^{\mathsf{T}}, \tag{3}$$

where we do not require the rank 1 terms to be symmetric (even if the block $A_t^{j,l}(r, s)$ itself is symmetric). Allowing for nonsymmetric matrices in the rank 1 decomposition is more general than what is done in [13, 17, 18], and as explained in Sect. 6.1 this can be important for performance.

We interpret (1) as the dual of the semidefinite program

$$\begin{aligned} &\text{minimize} && \sum_{j=1}^{J}\langle b^j, x^j\rangle \\ &\text{subject to} && \sum_{j=1}^{J}(B^j)^{\mathsf{T}} x^j = c \end{aligned}$$

$$X^j = \sum_{t \in T_j} x_t^j A_t^j - C^j \succeq 0, \qquad j = 1, \ldots, J, \tag{4}$$

where we optimize over the vectors of free variables x^j and the positive semidefinite block matrices $X^j = \operatorname{diag}(X^{j,1}, \ldots, X^{j,L_j})$.

Using the notation X for the block matrix $\operatorname{diag}(X^1, \ldots, X^J)$ and Y for the block matrix $\operatorname{diag}(Y^1, \ldots, Y^J)$, the duality gap for primal feasible (x, X) and dual feasible (y, Y) is given by

$$b^{\mathsf{T}} x - \langle C, Y \rangle - c^{\mathsf{T}} y = \langle X, Y \rangle.$$

We assume strong duality holds, so that if (x, X) and (y, Y) are optimal, then $\langle X, Y \rangle = 0$, and hence $XY = 0$.

The primal-dual algorithm starts with infeasible solutions (x, X) and (y, Y), where X and Y are positive definite. At each iteration, a Newton direction (dx, dX, dy, dY) is computed for the system of primal and dual linear constraints and the centering condition $XY = \beta\mu I$. Here μ is the surrogate duality gap $\langle X, Y \rangle$ divided by the size of the matrices, and β is a solver parameter between 0 and 1. Then (x, X, y, Y) is replaced by $(x+s dx, X+s dX, y+s dy, Y+s dY)$ for some step size s that ensures the matrices stay positive definite. Here we only discuss the process of finding the search direction since exploiting the special form (2) happens in this part of the algorithm. See [13] for more details on the remaining parts of the algorithm.

To compute the Newton search direction we replace the variables (x, X, y, Y) by $(x + dx, X + dX, y + dy, Y + dY)$ in the primal and dual constraints, which gives

$$X^j + dX^j = \sum_{t \in T_j} (x_t^j + dx_t^j) A_t^j - C^j, \tag{5}$$

$$\sum_{j=1}^{J} (B^j)^{\mathsf{T}} (x^j + dx^j) = c, \tag{6}$$

$$\left\langle A_*^j, Y^j + dY^j \right\rangle + B^j (y + dy) = b^j. \tag{7}$$

Then we apply the same substitution in the centering condition and linearize to get

$$X^j Y^j + X^j dY^j + dX^j Y^j = \mu I. \tag{8}$$

Substituting the expression for dX^j from (5) into (8) and then the expression for dY^j from (8) into (7) gives

$$\Big\langle A_*^j, Y^j + (X^j)^{-1} \Big(\mu I - X^j Y^j - \Big(\sum_{t \in T_j} (x_t^j + dx_t^j) A_t^j - C^j - X^j \Big) Y^j \Big) \Big\rangle$$
$$+ B^j (y + dy) = b^j.$$

Together with constraint (6) (which is responsible for the last row in the system) this can be written as the following linear system in dx and dy:

$$\begin{pmatrix} S^1 & 0 & \cdots & 0 & -B^1 \\ 0 & S^2 & \cdots & 0 & -B^2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & S^J & -B^J \\ (B^1)^\mathsf{T} & (B^2)^\mathsf{T} & \dots & (B^J)^\mathsf{T} & 0 \end{pmatrix} \begin{pmatrix} dx^1 \\ dx^2 \\ \vdots \\ dx^J \\ dy \end{pmatrix} = \begin{pmatrix} -b^1 - \langle A^1_*, Z^1 - Y^1 \rangle + B^1 y \\ -b^2 - \langle A^2_*, Z^2 - Y^2 \rangle + B^2 y \\ \vdots \\ -b^J - \langle A^J_*, Z^J - Y^J \rangle + B^J y \\ c - \sum_{j=1}^J (B^j)^\mathsf{T} x^j \end{pmatrix},$$

Here $Z^j = (X^j)^{-1}\big(\big(\sum_t x_t^j A_t^j - C^j\big)Y^j - \mu I\big)$ and the blocks S^j that form the Schur complement matrix $S = \operatorname{diag}(S^1, \dots, S^J)$ have entries

$$S^j_{ab} = \big\langle A^j_a, (X^j)^{-1} A^j_b Y^j \big\rangle.$$

The above system can be solved to obtain dx and dy. From this dX and dY can be computed, where instead of computing dY as $X^{-1}(\mu I - XY - dXY)$ we set

$$dY = \frac{X^{-1}(\mu I - XY - dXY) + (X^{-1}(\mu I - XY - dXY))^\mathsf{T}}{2}$$

so that Y stays symmetric.

In general, the computation of the Schur complement matrix S and solving the above linear system are the main computational steps.

Due to the clusters, the matrix S is block-diagonal, so that the Cholesky factorization $S = LL^\mathsf{T}$ can be computed blockwise. By using the decomposition

$$\begin{pmatrix} S & -B \\ B^\mathsf{T} & 0 \end{pmatrix} = \begin{pmatrix} L & 0 \\ B^\mathsf{T} L^{-\mathsf{T}} & I \end{pmatrix} \begin{pmatrix} I & 0 \\ 0 & B^\mathsf{T} L^{-\mathsf{T}} L^{-1} B \end{pmatrix} \begin{pmatrix} L^\mathsf{T} & -L^{-1}B \\ 0 & I \end{pmatrix},$$

we can solve the system by solving several triangular systems. The inner matrix $B^\mathsf{T} L^{-\mathsf{T}} L^{-1} B$ is positive definite, so we can again use a Cholesky decomposition.

Due to the low-rank constraint matrices, we can compute the blocks S^j more efficiently. Suppose for simplicity the constraint matrices are of the form

$$A_t^{j,l} = \sum_{r=1}^{\eta_t^{j,l}} \lambda_{t,r}^{j,l} v_{t,r}^{j,l} (w_{t,r}^{j,l})^\mathsf{T},$$

where $\eta_t^{j,l}$ is the rank of the matrix $A_t^{j,l}$. Then we can write

$$\begin{aligned}
S^j_{ab} &= \sum_{l=1}^{L_j} \langle A_a^{j,l}, (X^{j,l})^{-1} A_b^{j,l} Y^{j,l} \rangle \\
&= \sum_{l=1}^{L_j} \sum_{r_1=1}^{\eta_a^{j,l}} \sum_{r_2=1}^{\eta_b^{j,l}} \lambda^{j,l}_{a,r_1} \lambda^{j,l}_{b,r_2} \left\langle v^{j,l}_{a,r_1} (w^{j,l}_{a,r_1})^{\mathsf{T}}, (X^{j,l})^{-1} v^{j,l}_{b,r_2} (w^{j,l}_{b,r_2})^{\mathsf{T}} Y^{j,l} \right\rangle \\
&= \sum_{l=1}^{L_j} \sum_{r_1=1}^{\eta_a^{j,l}} \sum_{r_2=1}^{\eta_b^{j,l}} \lambda^{j,l}_{a,r_1} \lambda^{j,l}_{b,r_2} \left((w^{j,l}_{a,r_1})^{\mathsf{T}} (X^{j,l})^{-1} v^{j,l}_{b,r_2} \right) \left((w^{j,l}_{b,r_2})^{\mathsf{T}} Y^{j,l} v^{j,l}_{a,r_1} \right),
\end{aligned}$$

which shows we can compute S^j_{ab} efficiently by precomputing the bilinear pairings

$$(w^{j,l}_{a,r_1})^{\mathsf{T}} (X^{j,l})^{-1} v^{j,l}_{b,r_2} \quad \text{and} \quad (w^{j,l}_{b,r_2})^{\mathsf{T}} Y^{j,l} v^{j,l}_{a,r_1}.$$

In the implementation, we use similar techniques for the more general constraint matrices of the form (2). Since we use high-precision arithmetic, it is beneficial to use matrix-matrix multiplication with subcubic complexity, and therefore we compute the above pairings efficiently by first creating the matrices $V^{j,l}$ and $W^{j,l}$ with the columns $v^{j,l}_{a,r}$ and $w^{j,l}_{a,r}$, respectively, and then performing fast matrix multiplication to compute

$$(W^{j,l})^{\mathsf{T}} (X^{j,l})^{-1} V^{j,l} \quad \text{and} \quad (W^{j,l})^{\mathsf{T}} Y^{j,l} V^{j,l}.$$

Due to the block structures, the algorithm is relatively easy to parallelize. The best way to parallelize, however, depends on both the problem characteristics and the type of computing system used. The `SDPB` solver specializes in problems with large amounts of clusters with similar-sized blocks, which can be distributed over different nodes in a multi-node system in which there is communication latency between the nodes [13, 19].

Problems in discrete geometry typically consist of few clusters, and have a large variation in both the number of blocks per cluster and in the size of the blocks; see for example Sect. 6.1. The majority of the workload can be due to a single cluster, hence distributing clusters over nodes in a multi-node system is not a good parallelization strategy in this case. Instead, we focus on distributing the workload over multiple cores in a single-node shared-memory system.

Most of the matrix operations can be done block-wise. We distribute the blocks over the cores such that the workload for each core is about equal. Since the matrices in the products $(W^{j,l})^{\mathsf{T}} (X^{j,l})^{-1} V^{j,l}$ and $(W^{j,l})^{\mathsf{T}} Y^{j,l} V^{j,l}$ can be very large, we split these multiplications into several parts which we distribute over the cores.

3 Polynomial matrix programs

For univariate and multivariate problems, sums-of-squares characterizations, and in particular Putinar's characterization [27], are commonly used to model polynomial constraints as semidefinite constraints. Parrilo and Löfberg [15] show that by using sampling as opposed to coefficient matching we get a semidefinite programming formulation with low-rank constraint matrices. There has also been research into writing polynomial *matrix* constraints as semidefinite constraints [28, 29]. In this expository section, we consider the combination of multivariate polynomial matrix programs with the sampling approach and show precisely what kind of low-rank constraints appear when reducing these to semidefinite programs.

Let

$$\mathcal{S}(G) = \Big\{x \in \mathbb{R}^n : g(x) \geq 0 \text{ for all } g \in G\Big\}$$

be the semialgebraic set generated by a finite set of polynomials $G \subseteq \mathbb{R}[x]$ in n variables. Fix $m \in \mathbb{N}$ and define the quadratic module

$$\mathcal{M}(G) = \operatorname{cone}\Big\{gp^{\mathsf{T}}p : g \in G \cup \{1\},\ p \in \mathbb{R}[x]^{m\times m}\Big\}.$$

We say $\mathcal{M}(G)$ is Archimedean if for every $p \in \mathbb{R}[x]^{m\times m}$ there is a $c \in \mathbb{N}$ such that $cI - p^{\mathsf{T}}p \in \mathcal{M}(G)$. As shown in [29], this is equivalent to the condition that there is a $c \in \mathbb{N}$ such that $(c - \sum_i x_i^2)I \in \mathcal{M}(G)$. Intuitively, $\mathcal{M}(G)$ being Archimedean gives an algebraic certificate for the compactness of $\mathcal{S}(G)$. A polynomial matrix $f \in \mathbb{R}[x]^{m\times m}$ is said to be positive (semi)definite on $D \subseteq \mathbb{R}^n$ if the matrix $f(x)$ is positive (semi)definite for every $x \in D$. This is denoted by $f \succ 0$ ($f \succeq 0$) on D.

A polynomial matrix program is an optimization problem of the form

$$\begin{aligned} &\text{maximize} && \langle b, y\rangle \\ &\text{subject to} && P_0^j(x) + \sum_{i=1}^N y_i P_i^j(x) \succeq 0 \text{ on } \mathcal{S}(G_j), \qquad j = 1, \ldots, J, \end{aligned} \tag{9}$$

where we optimize over the vector $y \in \mathbb{R}^N$. The problem is defined by the sets $G_1, \ldots, G_J$ (where each of these sets may consist of polynomials in a different number of variables), the matrix polynomials P_i^j, and the vector b. In [13] the special case with $n = 1$ and $G_j = \{x\}$ is considered.

For the $m = 1$ case, positivity constraints on a set $\mathcal{S}(G)$ are modeled using weighted sums-of-squares polynomials. Such polynomials are trivially nonnegative on $\mathcal{S}(G)$, and by a theorem of Putinar [27] one can prove convergence when increasing the maximum degree of the sum-of-squares polynomials, assuming $\mathcal{M}(G)$ is Archimedean. A similar approach can be used for polynomial matrix programs. For this, we need a generalization of Putinar's theorem for matrix polynomials by Hol and Scherer [28] (with a different proof given by Klep and Schweighofer in [29]).

Theorem 1 [28, 29] *Let $f \in \mathbb{R}[x]^{m\times m}$ and $G \subseteq \mathbb{R}[x]$ finite. Suppose $\mathcal{M}(G)$ is Archimedean. If $f \succ 0$ on $\mathcal{S}(G)$, then $f \in \mathcal{M}(G)$.*

Similar to the non-matrix case, the requirement $f \succ 0$ can be weakened to $f \succeq 0$ when considering the univariate case where $\mathcal{S}(G)$ is $\mathbb{R}$, $\mathbb{R}_{\geq a}$, or $[a, b]$. In addition, $\mathcal{M}(G)$ is not required to be Archimedean in that case.

To state the relaxed problem, we consider the truncated quadratic module generated by G:

$$\mathcal{M}^d(G) = \operatorname{cone}\Big\{gp^{\mathsf{T}}p : g \in G \cup \{1\},\ p \in \mathbb{R}[x]^{m\times m},\ \deg(gp^{\mathsf{T}}p) \leq d\Big\}.$$

This gives

$$\begin{aligned} &\text{maximize} && \langle b, y\rangle \\ &\text{subject to} && P_0^j + \sum_{i=1}^N y_i P_i^j \in \mathcal{M}^d(G_j), && j = 1, \ldots, J. \end{aligned} \tag{10}$$

Let p^* and p_d^* denote the optimal values of problem (9) and (10), respectively. For all d we have $p_d^* \leq p_{d+1}^* \leq p^*$ and the following corollary whose proof is standard shows convergence under mild conditions.

Corollary 1 *Suppose* (9) *is strictly feasible and $\mathcal{M}(G_j)$ is Archimedean for every j. Then $p_d^* \to p^*$ as $d \to \infty$.*

It follows from the next lemma that we can model the elements in $\mathcal{M}^d(G)$ using positive semidefinite matrices. Let $b_d(x)$ be a vector whose elements form a basis for the polynomials of degree at most d.

Lemma 1 *For $f \in \mathbb{R}[x]^{m\times m}$ with $\deg(f) = 2d$ we have $f = p^{\mathsf{T}}p$ for some $p \in \mathbb{R}[x]^{t\times m}$ if and only if*

$$f = (b_d(x) \otimes I_m)^{\mathsf{T}} Y (b_d(x) \otimes I_m)$$

for some positive semidefinite matrix Y.

Proof Let δ be the length of $b_d(x)$. Then $p = Z(b_d(x) \otimes I_m)$ for some $Z \in \mathbb{R}^{t\times m\delta}$, and hence $p^{\mathsf{T}}p = (b(x) \otimes I_m)^{\mathsf{T}} Z^{\mathsf{T}} Z (b(x) \otimes I_m)$. Now note that $Y = Z^{\mathsf{T}}Z$ is positive semidefinite and any positive semidefinite matrix admits such a decomposition. □

The above lemma shows the elements of $\mathcal{M}^d(G_j)$ are of the form

$$\sum_{g\in G_j\cup\{1\}} g(x)(b_{d-\lfloor \deg(g)/2\rfloor}(x) \otimes I_{m_j})^{\mathsf{T}} Y_g^j (b_{d-\lfloor \deg(g)/2\rfloor}(x) \otimes I_{m_j}),$$

for positive semidefinite matrices Y_g^j. The entry on row r and column s is equal to

$$\sum_{g\in G_j\cup\{1\}} g(x)\Big\langle(b_{d-\lfloor \deg(g)/2\rfloor}(x) \otimes I_{m_j})^{\mathsf{T}} Y_g^j (b_{d-\lfloor \deg(g)/2\rfloor}(x) \otimes I_{m_j}),\, E_{r,s}^{m_j}\Big\rangle$$

$$= \sum_{g \in G_j \cup \{1\}} g(x) \left\langle Y_g^j, b_{d-\lfloor \deg(g)/2 \rfloor}(x) b_{d-\lfloor \deg(g)/2 \rfloor}(x)^{\mathsf{T}} \otimes E_{r,s}^{m_j} \right\rangle$$

where $E_{r,s}^{m_j} = e_r e_s^{\mathsf{T}}$ is the standard basis of $\mathbb{R}^{m_j \times m_j}$.

This leads to the optimization problem

$$\begin{array}{lll} \text{maximize} & \langle b, y \rangle & \\ \text{subject to} & P_0^j(x)_{r,s} + \sum_{i=1}^{N} y_i P_i^j(x)_{r,s} = \left\langle M^j(x)_{r,s}, Y^j \right\rangle, & j \in [J],\ r, s \in [m_j] \\ & Y^j \succeq 0, & j \in [J], \end{array}$$

where

$$M^j(x)_{r,s} = \bigoplus_{g \in G_j \cup \{1\}} g(x) b_{d-\lfloor \deg(g)/2 \rfloor}(x) b_{d-\lfloor \deg(g)/2 \rfloor}(x)^{\mathsf{T}} \otimes E_{r,s}^{m_j}.$$

In applications, the formulation (9) can be extended to a more general problem, where the polynomial matrices depend linearly on positive semidefinite matrix variables in addition to the free variables and where there are linear equality constraints on the free variables and the additional positive semidefinite matrix variables. We, therefore, use the following general form for a sums-of-squares problem:

$$\begin{array}{lll} \text{maximize} & \sum_{j=1}^{J} \langle C^j, Y^j \rangle + \langle b, y \rangle & \\ \text{subject to} & \left\langle A_*^j(x), Y^j \right\rangle + B^j(x) y = c^j(x), & j = 1, \ldots, J \\ & Y^j \succeq 0, & j = 1, \ldots, J. \end{array} \tag{11}$$

Here we use the notation

$$\langle A_*^j(x), Y^j \rangle = \big(\langle A_q^j(x), Y^j \rangle \big)_{q=1,\ldots,Q_j},$$

where

$$A_q^j(x) = \bigoplus_{l=1}^{L_j} \sum_{r,s=1}^{R_j(l)} A_q^{j,l}(r,s)(x) \otimes E_{r,s}^{R_j(l)},$$

and where the matrices $A_q^{j,l}(r,s)$ are of low rank. Here Q_j is the number of polynomial constraints in the j-th cluster, where different clusters are only linked via the free variables but not via the positive semidefinite matrix variables. Moreover, L_j specifies the number of blocks on the diagonal of $A_q^j(x)$, where the l-th block is a $R_j(l) \times R_j(l)$ block matrix.

In (11) we optimize over the free variables y and the positive semidefinite block-diagonal matrices Y^j. Here, the blocks Y_l^j do not all have to correspond to sum-of-squares polynomial matrices, and not all constraints have to be polynomial constraints (that is, some constraint can use degree 0 polynomials). See the examples in Sect. 6.

The usual approach for converting a problem of the form (11) to a semidefinite program is to equate the coefficients of the polynomials in a common basis. This potentially gives sparsity but destroys the low-rank structure of the matrices. Instead, we use the sampling approach as introduced by Löfberg and Parrilo in [15] and used by Simmons-Duffin in [13]. For each $1 \leq j \leq J$ and $1 \leq q \leq Q_j$ we define a unisolvent set of points M_q^j for the polynomial subspace spanned by the entries of $A_q^j(x)$, the entries in the q-th row of $B^j(x)$, and the q-th entry of $c^j(x)$. This is a set of points such that any polynomial in this space which is zero on M_q^j is identically zero. Then we consider the linear constraints

$$\left\langle A_q^j(x'), Y^j \right\rangle + (B^j(x')y)_q = c_q^j(x')$$

for $x' \in M_q^j$. That is, when going from (11) to (1), we set

$$T_j = \{(q, x') : q = 1, \ldots, Q_j,\ x' \in M_q^j\}.$$

4 Combining symmetry reduction and sampling

In this section, we investigate the combination of symmetry reduction with sampling as opposed to coefficient matching. For this, we first give an exposition of the symmetry reduction approach for polynomial optimization by Gatermann and Parrilo [30], where we give more details for the constructions important in this paper. For notational simplicity we consider only polynomial programs as opposed to matrix polynomial programs.

Suppose the polynomials $P_0, \ldots, P_N$ and the set $\mathcal{S}(G)$ are invariant under the action of a finite group Γ, and assume the polynomials in G are chosen to be Γ-invariant (which is in fact always possible if $\mathcal{S}(G)$ is invariant under Γ; see "Appendix A"). Then the sums-of-squares characterization for a constraint of the form

$$P_0 + \sum_{i=1}^N y_i P_i \geq 0 \text{ on } \mathcal{S}(G)$$

can be written more efficiently.

Let Γ be a finite group with a linear action on $\mathbb{C}^n$, and define the representation $L\colon \Gamma \to \mathrm{GL}(\mathbb{C}[x])$ by $L(\gamma)p(x) = p(\gamma^{-1}x)$. Here $\mathrm{GL}(\mathbb{C}[x])$ is the automorphism group of the vector space $\mathbb{C}[x]$. Let $\widehat{\Gamma}$ be a complete set of irredicible representations (π, V_π) of Γ. By Maschke's theorem, we get the decomposition

$$\mathbb{C}[x] = \bigoplus_{\pi \in \widehat{\Gamma}} \bigoplus_i H_{\pi,i},$$

where $H_{\pi,i}$ is equivalent to V_π. Since the space of homogeneous polynomials of degree k is invariant under the action of Γ, we may assume that $H_{\pi,i}$ is spanned by homogeneous polynomials of the same degree.

Since Γ is finite we can choose a basis $e_{\pi,1}, \ldots, e_{\pi,d_\pi}$ of V_π in which the linear operators $\pi(\gamma)$ are unitary matrices. We want to define bases $e_{\pi,i,1}, \ldots, e_{\pi,i,d_\pi}$ of $H_{\pi,i}$ which are symmetry adapted in the sense that the restriction of $L(\gamma)$ to the invariant subspace $H_{\pi,i}$ in this basis is exactly $\pi(\gamma)$.

Such a basis exists since $H_{\pi,i}$ is equivalent to V_π, so there are Γ-equivariant isomorphisms $T_{\pi,i} \colon V_\pi \to H_{\pi,i}$ and we can define $e_{\pi,i,j} = T_{\pi,i} e_{\pi,j}$. Then it follows that $L(\gamma)e_{\pi,i,j} = \sum_k \pi(\gamma)_{k,j} e_{\pi,i,k}$. As described in [31, Section 2.7] a symmetry-adapted basis can be constructed by defining the operators

$$p^\pi_{j,j'} = \frac{d_\pi}{|\Gamma|} \sum_{\gamma \in \Gamma} \pi(\gamma^{-1})_{j',j} L(\gamma), \tag{12}$$

and then choosing bases $\{e_{\pi,i,1}\}_i$ of $\mathrm{Im}(p^\pi_{1,1})$ and setting $e_{\pi,i,j} = p^\pi_{j,1} e_{\pi,i,1}$. Then,

$$\begin{aligned} L(\tilde{\gamma})e_{\pi,i,j} &= \frac{d_\pi}{|\Gamma|} \sum_{\gamma \in \Gamma} \pi(\gamma^{-1})_{1,j} L(\tilde{\gamma}\gamma) e_{\pi,i,1} = \frac{d_\pi}{|\Gamma|} \sum_{\gamma \in \Gamma} \pi(\gamma^{-1}\tilde{\gamma})_{1,j} L(\gamma) e_{\pi,i,1} \\ &= \frac{d_\pi}{|\Gamma|} \sum_{\gamma \in \Gamma} \sum_{k=1}^{d_\pi} \pi(\gamma^{-1})_{1,k} \pi(\tilde{\gamma})_{k,j} L(\gamma) e_{\pi,i,1} = \sum_{k=1}^{d_\pi} \pi(\tilde{\gamma})_{k,j} e_{\pi,i,k}. \end{aligned}$$

If the irreducible representations occurring in the decomposition of $\mathbb{C}[x]$ are of real type, then we can choose bases so that the matrices $\pi(\gamma)$ are orthogonal. If moreover $\mathbb{R}[x]$ is Γ-invariant, then the above explicit construction shows we can take the symmetry adapted basis to be real: Since (12) is a real operator, we can choose a real basis $\{e_{\pi,i,1}\}_i$ of the image of $p^\pi_{1,1}$ and $e_{\pi,i,j} = p^\pi_{j,1} e_{\pi,i,1}$ will be real as well.

For each π we define the matrix polynomial E_π by

$$E_\pi(x)_{i,i'} = \frac{1}{d_\pi} \sum_{j=1}^{d_\pi} e_{\pi,i,j}(x) \overline{e_{\pi,i',j}(x)}$$

That the matrices $E_\pi(x)$ are Γ-invariant follows from the alternative definition

$$E_\pi(x)_{i,i'} = \frac{1}{|\Gamma|} \sum_{\gamma \in \Gamma} e_{\pi,i,j}(\gamma^{-1}x) \overline{e_{\pi,i',j}(\gamma^{-1}x)}$$

(for any $1 \le j \le d_\pi$), which follows from

$$\frac{1}{|\Gamma|} \sum_{\gamma \in \Gamma} e_{\pi,i,j}(\gamma^{-1}x) \overline{e_{\pi,i',j}(\gamma^{-1}x)}$$

$$\begin{aligned}
&= \frac{1}{|\Gamma|} \sum_{\gamma \in \Gamma} \sum_{l=1}^{d_\pi} \pi(\gamma^{-1})_{l,j} e_{\pi,i,l}(x) \sum_{k=1}^{d_\pi} \overline{\pi(\gamma^{-1})_{k,j} e_{\pi,i',k}(x)} \\
&= \sum_{l,k=1}^{d_\pi} e_{\pi,i,l}(x) \overline{e_{\pi,i',k}(x)} \frac{1}{|\Gamma|} \sum_{\gamma \in \Gamma} \pi(\gamma^{-1})_{l,j} \overline{\pi(\gamma^{-1})_{k,j}} \\
&= \sum_{l,k=1}^{d_\pi} e_{\pi,i,l}(x) \overline{e_{\pi,i',k}(x)} \, \frac{\delta_{kl}}{d_\pi},
\end{aligned}$$

where we use the Schur orthogonality relations (see, e.g., [31, Section 2.2]) in the last equality.

For $d \in \mathbb{N}$, we define $E_\pi^d(x)$ as the submatrix of $E_\pi(x)$ indexed by rows and columns for which $\deg e_{\pi,i,j}(x) \leq d$. The following proposition shows how these matrices can be used to parametrize Hermitian sum-of-squares polynomials by Hermitian positive semidefinite matrices. If the symmetry-adapted basis is real, then the matrices E_π^d are symmetric and we can parametrize real sum-of-squares polynomials by positive semidefinite matrices.

Proposition 1 *If $p \in \mathbb{C}[x]_{\leq 2d}$ is a Γ-invariant Hermitian sum-of-squares polynomial, then there are Hermitian positive semidefinite matrices C_π such that*

$$p(x) = \sum_\pi \left\langle C_\pi, E_\pi^d(x) \right\rangle.$$

Proof Define the column vector $b(x) = (e_{\pi,i,j}(x))_{\pi,i,j}$ with $\deg e_{\pi,i,j}(x) \leq d$. Since p is a Hermitian sum-of-squares polynomial it can be written as

$$p(x) = \sum_i (c_i^* b(x))^* (c_i^* b(x)),$$

so there exists a Hermitian positive semidefinite matrix A such that

$$p(x) = b(x)^* A b(x).$$

Let $\rho(\gamma)$ be the matrix obtained by expressing the restriction of $L(\gamma)$ to $\mathbb{C}[x]_{\leq 2d}$ in the symmetry adapted basis, so that

$$\rho(\gamma) = \bigoplus_\pi I_{m_\pi} \otimes \pi(\gamma) \tag{13}$$

and $\rho(\gamma) b(x) = b(\gamma^{-1} x)$ for all x and γ. Define

$$B = \frac{1}{|\Gamma|} \sum_{\gamma \in \Gamma} \rho(\gamma)^* A \rho(\gamma).$$

Then $\rho(\gamma)^* B \rho(\gamma) = B$ for all $\gamma \in \Gamma$ and

$$p(x) = \frac{1}{|\Gamma|} \sum_{\gamma \in \Gamma} b(\gamma^{-1}x)^* A b(\gamma^{-1}x) = b(x)^* B b(x).$$

By writing B in the block form $B = (B_{(\pi,i),(\pi',i')})$ and using (13) we get

$$\pi(\gamma) B_{(\pi,i),(\pi',i')} = B_{(\pi,i),(\pi',i')} \pi'(\gamma)$$

for all $\gamma \in \Gamma$. By Schur's lemma $B_{(\pi,i),(\pi',i')}$ is a multiple of the identity if $\pi = \pi'$ and zero otherwise (see, e.g., [31, Section 2.2]). This shows $B = \bigoplus_\pi \frac{1}{d_\pi} C_\pi \otimes I_{d_\pi}$ for positive semidefinite matrices C_π. We then have

$$\begin{aligned} p(x) &= \langle B, b(x)b(x)^* \rangle \\ &= \left\langle \bigoplus_\pi \frac{1}{d_\pi} C_\pi \otimes I_{d_\pi}, b(x)b(x)^* \right\rangle \\ &= \sum_\pi \sum_{j=1}^{d_\pi} \sum_{i,i'=1}^{m_\pi} \frac{1}{d_\pi} (C_\pi)_{i,i'} e_{\pi,i,j}(x) e_{\pi,i',j}(x)^* \\ &= \sum_\pi \left\langle C_\pi, E_\pi^d(x) \right\rangle, \end{aligned}$$

which completes the proof. □

In applications, the symmetry groups often are reflection groups (see, e.g., [8] and Sect. 6.1), and as described in [30] for these groups we can choose the symmetry adapted basis in such a way that the matrices $E_\pi(x)$ have a tensor structure. By, e.g., [32, Section 3.6], $\mathbb{C}[x]$ is a free module over the invariant ring $\mathbb{C}[x]^\Gamma$ of rank $|\Gamma|$. Moreover, the span of any $\mathbb{C}[x]^\Gamma$ module basis of $\mathbb{C}[x]$ is equivalent to the regular representation of Γ. This means we have the decomposition

$$\mathbb{C}[x] = \mathbb{C}[x]^\Gamma \bigoplus_\pi \bigoplus_{i=1}^{d_\pi} V_{\pi,i},$$

where $V_{\pi,i} \subseteq \mathbb{C}[x]$ is equivalent to V_π. As before, we may assume that $V_{\pi,i}$ is spanned by homogeneous polynomials of the same degree.

Let $\{f_{\pi,i,j}\}$ be a symmetry adapted basis of

$$\bigoplus_\pi \bigoplus_{i=1}^{d_\pi} V_{\pi,i}.$$

Then we can choose the symmetry-adapted basis of $\mathbb{C}[x]$ to be of the form

$$e_{\pi,(i,k),j}(x) = f_{\pi,i,j}(x)\, w_k(x),$$

where $w_k(x)$ is a basis of $\mathbb{C}[x]^\Gamma$. The matrix $E_\pi(x)$ can therefore be written as

$$E_\pi(x)_{(i,k),(i',k')} = \sum_{j=1}^{d_\pi} f_{\pi,i,j}(x) w_k(x) \overline{f_{\pi,i',j}(x) w_{k'}(x)},$$

i.e.,

$$E_\pi(x) = \Pi_\pi(x) \otimes w(x)w(x)^*,$$

where $\Pi_\pi(x)$ is the matrix given by $\Pi_\pi(x)_{i,i'} = \sum_{j=1}^{d_\pi} f_{\pi,i,j}(x)\overline{f_{\pi,i',j}(x)}$.

From now on we assume that we can and do choose the symmetry-adapted basis to be real so that the matrices $E_\pi(x)$ are symmetric. We then replace the constraint

$$P_0 + \sum_{i=1}^{N} y_i P_i \succeq 0 \text{ on } \mathcal{S}(G)$$

by the condition that there are positive semidefinite matrices $Y_{g,\pi}$ for which

$$P_0(x) + \sum_{i=1}^{N} y_i P_i(x) = \sum_{g \in G \cup \{1\}} g(x) \sum_\pi \left\langle Y_{g,\pi}, E_\pi^{d-\lfloor \deg(g)/2 \rfloor}(x) \right\rangle. \tag{14}$$

As in Sect. 3 we want to model constraint (14) by the linear constraints obtained from evaluating it at a unisolvent set of points. Since P_i, g, and E_π are all Γ-invariant, it is sufficient to consider a unisolvent set for the subspace of Γ-invariant polynomials of degree at most $2d$. A unisolvent set is said to be minimal if any strict subset is not unisolvent. In the following proposition we show that the constraints arising from evaluating (14) at a minimal unisolvent set are linearly independent, which is essential for the solver.

Proposition 2 *Evaluating* (14) *at a minimal unisolvent set M for the subspace of Γ-invariant polynomials of degree at most $2d$ yields $|M|$ linearly independent constraints in the variables of the optimization problem.*

Proof By the proof of Proposition 1, we can express any polynomial as a linear combination of the entries of the matrix $\oplus_\pi E_\pi^d$, which means that these entries span the space of Γ-invariant polynomials of degree at most $2d$.

In particular, there is a subset $\{b_j(x)\}_j$ of the entries of $\oplus_\pi E_\pi^d$ which forms a basis for the invariant polynomials. Since the unisolvent set M is minimal, the vectors $(b_j(x'))_j$ are linearly independent for $x' \in M$. This implies the matrices $\oplus_\pi E_\pi^d(x')$ are linearly independent, and hence the linear combinations

$$\sum_\pi \left\langle Y_{1,\pi}, E_\pi^d(x') \right\rangle,$$

for $x' \in M$, are linearly independent. This shows that evaluating (14) on M yields $|M|$ linearly independent constraints. □

A symmetric polynomial optimization problem often arises as the symmetrization of a non-symmetric problem. For this reason, we want to know when and how a minimal unisolvent set for the non-symmetric problem gives a minimal unisolvent set for the symmetrized problem. The following lemma gives a sufficient condition for this to be the case.

Lemma 2 *Suppose M is a Γ-invariant, minimal unisolvent set for $\mathbb{R}[x]_{\leq 2d}$. Then any set of representatives R for the orbits in M is minimal unisolvent for $\mathbb{R}[x]^{\Gamma}_{\leq 2d}$.*

Proof Suppose $p \in \mathbb{R}[x]^{\Gamma}_{\leq 2d}$ satisfies $p(r) = 0$ for every $r \in R$. Then for every $x \in M$ there are $r \in R$ and $\gamma \in \Gamma$ with $x = \gamma r$, so $p(x) = p(\gamma r) = p(r) = 0$. Hence $p = 0$ by unisolvence of M. So R is unisolvent for $\mathbb{R}[x]^{\Gamma}_{\leq 2d}$.

Consider the projection operator $\mathcal{P}\colon \mathbb{R}[x]_{\leq 2d} \to \mathbb{R}[x]_{\leq 2d}$ defined by

$$\mathcal{P}p(x) = \frac{1}{|\Gamma|} \sum_{\gamma \in \Gamma} p(\gamma^{-1} x).$$

Let $b(x)$ be a column vector where the first entries form a basis for the eigenspace with eigenvalue 1 of $\mathcal{P}$ and the remaining entries form a basis for the kernel of $\mathcal{P}$. Consider the Vandermonde matrix $V = (b(x))_{x \in M}$, where M indexes the columns. By minimal unisolvence of M, the matrix V is square and nonsingular. We can perform invertible column operations to transform the first $|R|$ columns into $|\Gamma r|^{-1} \sum_{x \in \Gamma r} b(x)$, where Γr is the orbit represented by $r \in R$. Now note that the first $\dim \mathbb{R}[x]^{\Gamma}_{\leq 2d}$ entries in each column stay the same and the other entries in the first $|R|$ columns become 0. That is, after performing these column operations the matrix is of the form

$$V' = \begin{pmatrix} V_{11} & V_{12} \\ 0 & V_{22} \end{pmatrix}.$$

Note that V_{11} must be square: R is unisolvent, so the number of columns $|R|$ is at least the number of rows $\dim \mathbb{R}[x]^{\Gamma}_{\leq 2d}$, and V' is nonsingular. Hence $|R| = \dim \mathbb{R}[x]^{\Gamma}_{\leq 2d}$, i.e., R is minimal unisolvent. □

There are group actions for which a Γ-invariant, minimal unisolvent set M does not exist. Since Γ is a finite group, there are a finite number of orbit sizes o_i. Suppose M and R are as in Lemma 2. Then M can be decomposed into orbits such that there are k_i orbits with orbit size o_i. This directly means that $\sum_i k_i o_i = |M| = \dim \mathbb{R}[x]_{\leq 2d}$ because M is minimal unisolvent. Furthermore, the minimial unisolvence of R implies that $\sum_i k_i = |R| = \dim \mathbb{R}[x]^{\Gamma}_{\leq 2d}$. When this system of equations does not have a nonnegative integer solution, there does not exist a Γ-invariant, minimal unisolvent set. Moreover, even if the system does have a nonnegative integer solution, it is possible that there does not exist an invariant, minimal unisolvent set M. Such an example can be found in “Appendix B”.

In the following lemma we show that for a group action permuting $n + 1$ affinely independent vectors (which includes the important case of permuting coordinates), an invariant, minimal unisolvent set exists. For this we use a geometric criterion of minimal unisolvence by Chung and Yao [33]: a set M of size $\dim \mathbb{R}[x]_{\leq d}$ is minimal unisolvent for $\mathbb{R}[x]_{\leq d}$ if for every $x \in M$ there are hyperplanes $H_{x,1}, \dots, H_{x,d}$ such that

$$M \cap \left(\bigcup_{l=1}^{d} H_{x,l} \right) = M \setminus \{x\}.$$

Lemma 3 *Let S_{n+1} be the symmetric group on $n + 1$ elements. Suppose $\Gamma \subseteq S_{n+1}$ acts on $\mathbb{R}^n$ by permuting $n + 1$ affinely independent vectors $v_1, \dots, v_{n+1}$. Then there is a Γ-invariant, minimal unisolvent set M for $\mathbb{R}[x]_{\leq d}$.*

Proof Without loss of generality we can take $\Gamma = S_{n+1}$. For each $x \in \mathbb{R}^n$ there are unique coefficients $\alpha_1^x, \dots, \alpha_{n+1}^x$ such that $\sum_i \alpha_i^x = 1$ and $x = \sum_i \alpha_i^x v_i$. Let M be the set of points x for which

$$\alpha_1^x, \dots, \alpha_{n+1}^x \in \left\{ \frac{k}{d} : k = 0, \dots, d \right\}.$$

Note that M is invariant under the action of Γ.

For each $x \in M$ we define the hyperplanes

$$H_{x,(j,k)} = \left\{ y \in \mathbb{R}^n : d\alpha_j^y = k \right\}$$

for $1 \leq j \leq n + 1$ and $0 \leq k < d\alpha_j^x$. Here $H_{x,(j,k)}$ is a hyperplane because $H_{x,(j,k)} - k/dv_j$ is the affine hull of the vectors $(1 - k/d)v_i$ with $i \neq j$. Note that this gives $\sum_j d\alpha_j^x = d$ hyperplanes for each x.

Since $k < d\alpha_j^x$ we have $x \notin H_{x,(j,k)}$. Moreover, for any $y \in M \setminus \{x\}$ there is an i such that $\alpha_i^y < \alpha_i^x$, i.e., $y \in H_{x,(i,d\alpha_i^y)}$. So the geometric characterization is satisfied, hence M is minimal unisolvent. □

5 Computing good sample points and bases

To model the constraints of the form (14) using sampling we need to find a minimal unisolvent set for the space of Γ-invariant polynomials of degree at most $2d$ (where Γ is the trivial group if there is no symmetry). In Sect. 4 we explain when such a set can be derived from a minimal unisolvent set for $\mathbb{R}[x]_{\leq d}$. For the interior point method, however, we do not just want the set to be minimal unisolvent, but also to have good numerical conditioning. Moreover, as explained in Sect. 4, if the group is a reflection group (which is often the case in practice), then the matrix $E_\pi^d(x)$ in (14) is a submatrix of

$$\Pi_\pi(x) \otimes w(x)w(x)^{\mathsf{T}},$$

where for w we can choose any vector whose entries form a basis for the Γ-invariant polynomials of degree at most d. Note that in the case of no symmetry, $\Pi_\pi(x)$ is just the 1×1 identity matrix.

In [34] Sommariva and Vianello discuss a greedy method for finding a good sample set and a good basis for quadrature problems. In this section we adapt this to find a good minimal unisolvent set of sample points as well as a good basis for the entries of $w(x)$.

The space of polynomials corresponding to a constraint in the polynomial matrix program (9) is given by

$$W = \mathbb{R}[x]_{\leq 2d}^{\Gamma_j}.$$

Let v be a vector whose entries form a basis of W such that $\deg(v_k)$ is nondecreasing in k. Let M be a set containing m distinct points $x_1, \dots, x_m$ from the semialgebraic set $\mathcal{S}(G)$, where m is at least $\dim(W)$. Here the idea is that we take m much larger than $\dim(W)$ and later select a subset of good points. The Vandermonde matrix V with respect to v and M is defined by $V_{lk} = v_k(x_l)$, for $l = 1, \dots, m$ and $k = 1, \dots, \dim(W)$. The set M is unisolvent for W if and only if the kernel of V is trivial, and minimal unisolvent if additionally V is square.

The Fekete points for a compact domain and a polynomial space are defined as the points in the domain that maximize the determinant of the Vandermonde matrix in absolute value. Because a basis change multiplies the determinant by a constant not depending on the samples, the Fekete points do not depend on the basis. Instead of computing the Fekete points, which is hard to do in general [35], Sommariva and Vianello select a subset of a set of candidate points which approximately maximizes the determinant, the approximate Fekete points. This can be done greedily by a QR factorization of V^{T} with column pivoting; the points corresponding to the first $\dim(W)$ pivots approximately maximize the determinant. We now let M' be the subset of M corresponding to the first $\dim(W)$ many pivots, and let V' be the square submatrix of V by selecting the corresponding rows. If the original set M was unisolvent, then the determinant of V will be nonzero.

Because we use sample points to express the sums-of-squares constraints as semidefinite constraints, it is desirable for the numerical conditioning that the entries of w are orthogonal with respect to the chosen sample set M'. Such a basis can be obtained by a QR factorization of the Vandermonde matrix V', as also mentioned by Löfberg and Parrilo [15]. Here the columns of Q represent a new basis of W, where each basis element defines a polynomial by its evaluations on the sample set M'. For the polynomials w_i we now choose the polynomials in this basis which have degree at most d.

Note that for the implementation it is not necessary to recover the coefficients of the polynomials w_i in the monomial basis since we only ever use the evaluations of (14) on the points in M'. This means we can directly use the columns of Q (which are in fact the coefficients with respect to the Lagrange basis for the set M'). In fact, we extend the computer algebra system `AbstractAlgebra.jl` [36] we use in the interface to our solver with a new type called `SampledMPolyElem`, which is a polynomial defined only by its evaluations on a given set. This is very useful for modeling for instance the right-hand side of (14), where we have a mixture of polynomials (the weights $g(x)$ and the entries of the matrices $\Pi_\pi(x)$) and sampled polynomials (the entries of $w(x)$).

$* \quad * \quad *$

Since the matrices to which the linear algebra routines described above need to be applied are much larger than the matrices considered in the solver, we want to perform the operations in machine precision as much as possible (otherwise the preprocessing may become more expensive than solving the semidefinite program). Here we have to be careful that after performing the QR factorization of V', the degrees of the polynomials corresponding to the columns in Q are still nondecreasing in the column index, because we need the first columns to correspond to a basis of the polynomials in W of degree at most d. We, therefore, have to adapt [34, Algorithm 2] to our setting; this adaptation is also implemented in the package `ClusteredLowRankSolver.jl`.

First, we improve the basis by computing the QR factorization of V in machine precision. We then compute the matrix VR^{-1} using high-precision arithmetic and replace V with it. By using high-precision arithmetic we ensure that the degrees of the polynomials corresponding to the columns of V will still be nonincreasing in the column index. Although the columns of the new V will not be orthogonal (due to numerical issues in the QR factorization), they will be more orthogonal than before. We can optionally repeat this process a few times.

We then compute a pivoted QR factorization of V^T in machine precision, and as described above use it to select a suitable set of samples and define V' by selecting the corresponding rows of V.

To find a basis that is orthogonal with respect to this sample set we compute the QR factorization of V' in machine precision and compute $V'R^{-1}$ in high-precision arithmetic. Since the numerical conditioning of V' should now be relatively good, the columns of $V'R^{-1}$ will indeed be near orthogonal. We then select appropriate columns to use as basis polynomials for the entries of w, where as described above we do not need to compute the coefficients in the monomial basis.

6 Applications

In this section we consider two applications from discrete geometry: the kissing number problem and the binary sphere packing problem. The first application showcases the speed of the solution approach for a symmetric multivariate polynomial optimization problem. The second application showcases the stability of the sampling approach for a univariate polynomial matrix problem.

6.1 Kissing number problem

A subset C of the unit sphere $S^{n-1} = \{v \in \mathbb{R}^n : \langle v, v\rangle = 1\}$ is a spherical θ-code if $\langle v, v'\rangle \leq \cos(\theta)$ for all distinct $v, v' \in C$. In discrete geometry we are interested in the maximum size $A(n, \theta)$ of such a set. For $\cos\theta = 1/2$ this is the kissing number problem, where we ask for the maximum number of unit spheres that can simultaneously touch a central unit sphere. The kissing number problem has a rich history; see [37] for background information.

In [3], Bachoc and Vallentin introduce a three-point semidefinite programming bound that gives many of the best-known upper bounds on $A(n, \theta)$. Here we give the formulation of this bound leaving out an ad-hoc 2×2 matrix which does not contribute numerically [38].

Define the matrices $Y_k^n(u, v, t)$ by

$$Y_k^n(u, v, t)_{ij} = u^i v^j (1-u^2)^{k/2}(1-v^2)^{k/2} P_k^{n-1}\left(\frac{t-uv}{\sqrt{(1-u^2)(1-v^2)}}\right),$$

where P_k^n is the k-th degree Gegenbauer polynomial with parameter $n/2 - 1$, normalized such that $P_k^n(1) = 1$. Define the matrices $\bar{Y}_k^n(u, v, t)$ by

$$\bar{Y}_k^n(u, v, t) = \frac{1}{6}\sum_{\sigma \in S_3} \sigma Y_k^n(u, v, t),$$

where $\sigma \in S_3$ acts on Y_k^n by permuting its arguments. With these matrices the three-point bound is the problem

$$\begin{aligned}
&\text{minimize} && 1 + \sum_{k=0}^{2d} a_k + \langle \bar{Y}_0^n(1,1,1), F_0 \rangle && \\
&\text{subject to} && \sum_{k=0}^{2d} a_k P_k^n(u) + 3\sum_{k=0}^{d} \langle \bar{Y}_k^n(u,u,1), F_k \rangle \le -1, && u \in [-1, \cos\theta] \\
& && \sum_{k=0}^{d} \langle \bar{Y}_k^n(u,v,t), F_k \rangle \le 0, && (u,v,t) \in \mathcal{S}(G) \\
& && a_0, \ldots, a_{2d} \ge 0,\ F_0, \ldots, F_d \succeq 0, &&
\end{aligned}$$

where

$$\mathcal{S}(G) = \left\{(u, v, t) : -1 \le u, v, t \le \cos\theta,\ 1 + 2uvt - u^2 - v^2 - t^2 \ge 0\right\}.$$

Note that the multivariate constraints are symmetric under the action of the symmetric group S_3 on three elements, so that we use the techniques from Sects. 4 and 5.

We can view the above problem as being an extension of (9) with the additional positive semidefinite matrix variables a_k of size 1 and F_k of size $d - k + 1$. Using the approach from Sect. 3 we then obtain a problem in the form (11) and after sampling a semidefinite program of the form (1). We consider two ways of doing this.

In the first approach we view the problem as an extension of (9) without free variables, but where the polynomial constraints depend linearly on a_k and F_k. After converting to (11), the semidefinite program has positive semidefinite matrix variables

a_k, F_k, and variables corresponding to the sum-of-square polynomials. When going from (11) to (1) we have to evaluate the polynomials P_k^n, the polynomial matrices $\bar{Y}_k^n$, and the matrices arising from the sum-of-squares polynomials at the samples. Here it is beneficial to factor $\bar{Y}_k^n$ symbolically before evaluating it at the samples. The matrices $\bar{Y}_k^n(u, v, t)$ have rank at most 3 after evaluation at a point (u, v, t). It is however not clear whether we can write $\bar{Y}_k^n(u, v, t)$ as a sum of three symmetric rank 1 matrix polynomials of the form $\lambda(x)v(x)v(x)^{\mathsf{T}}$. We, therefore, decompose $\bar{Y}_k^n(u, v, t)$ as a sum of nonsymmetric rank 1 matrix polynomials and we use the fact that our semidefinite programming solver supports nonsymmetric rank 1 factors in the symmetric constraint matrices. For large d this will be the fastest approach.

For intermediate d we consider the alternative approach where we use free variables for the entries of a_k and F_k, write the polynomial constraints in terms of these free variables, and use additional rank 1 linear constraints to link the free variables to newly introduced positive semidefinite matrices a_k' and F_k'. This approach can be faster for small and intermediate d and uses less memory, which can be important for practical computations. For the computations discussed below (where d is at most 20) we use this approach. For more complicated problems, with for instance polynomial inequality constraints on unions of basic semialgebraic sets, this approach may be very useful since it allows more fine-grained clustering of the positive semidefinite matrices which the solver can exploit.

Initial computations for the three-point bound were performed by Bachoc and Vallentin using CSDP [39], but since this is a machine precision solver it was not possible to go beyond $d = 10$ [3]. Mittelman and Vallentin [4] then used the high precision solvers SDPA-QD and SDPA-GMP to perform computations up to $d = 14$. Later Machado and Oliviera [5] applied symmetry reduction and used SDPA-GMP to compute bounds up to degree $d = 16$.

For $d = 16$ and using the same symmetry reduction, our solver gives a speedup by a factor of 28 over the approach using SDPA-GMP using 8 cores, and a speedup by a factor of 9.6 using 1 core. Here we use the same hardware and the default settings of SDPA-GMP, except that we use 256 bits of floating point precision for all three-point bound computations.

It would be interesting to compare this to timings one would obtain using direct optimization over sum-of-squares polynomials, as developed by Skajaa, Ye, Papp, and Yıldız [40–42]. However, for this their approach would first have to be extended to semidefinite programs with polynomial constraints, and a high-precision solver would have to be implemented.

As can be seen in Figs. 1 and 2 the factor by which our solver is faster than the approach with SDPA-GMP increases with d. For the approach using SDPA-GMP the computation time theoretically scales as d^{12} when sparsity is not exploited; in practice we see that it scales as $d^{10.1}$. With our approach the computation time theoretically scales as d^9, and in practice we observe a scaling of $d^{8.55}$. The discrepancy between theory and practice for our approach can in part be explained by the fact that matrix multiplication in Arb is faster than cubic [43].

Because of the speedup of our approach we can perform computations up to $d = 20$ within a reasonable time frame (extrapolating Fig. 2 shows that the approach using

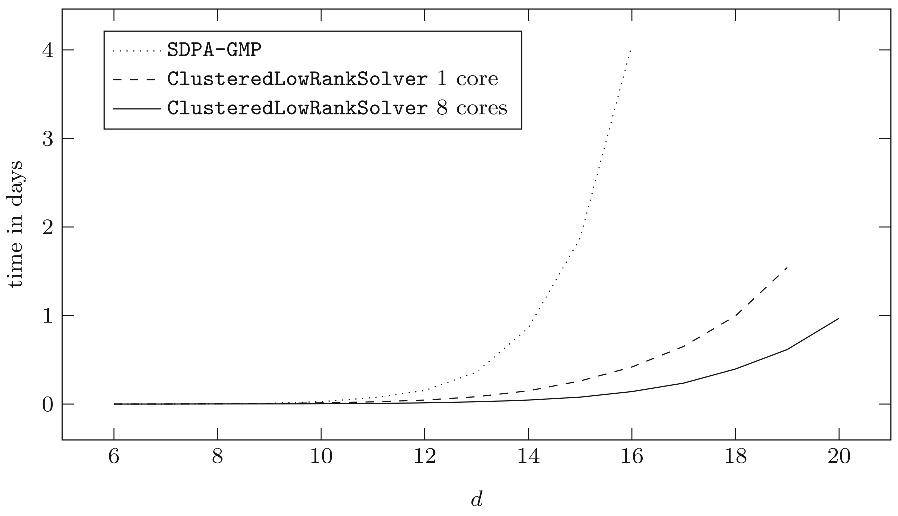

Fig. 1 The time needed to compute the three-point bound for the kissing number in dimension $n = 4$ for several degrees d on a linear scale, using SDPA-GMP and ClusteredLowRankSolver

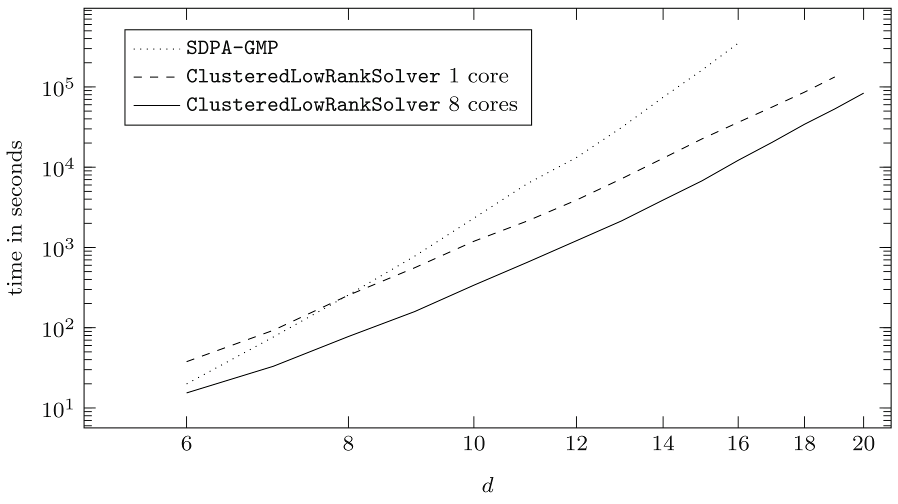

Fig. 2 The time needed to compute the three-point bound for the kissing number in dimension $n = 4$ for several degrees d on a log-log scale, using SDPA-GMP and ClusteredLowRankSolver

SDPA-GMP would have been slower by a factor 40 for $d = 20$). In Table 1 we show the kissing number bounds for $d = 16, \ldots, 20$ for dimensions up to 24. Dimension 2, 8, and 24 are omitted since the linear programming bound is sharp in these dimensions. After rounding down to the nearest integer, this improves the best known upper bounds in dimensions 11 through 23.

Rigorous verification of these bounds can be done using standard interval-arithmetic techniques (see, e.g., [5, 7]), the only caveat being that we first need to apply the reverse basis transformation obtained in Sect. 5 to the obtained solution. We did not perform

Table 1 Three-point bounds for the kissing number problem in dimensions 3-23

n	Lower bound	d	Upper bound	n	Lower bound	d	Upper bound
3	12	16	12.368580	14	1606	16	3177.7812
		17	12.364503			17	3176.4354
		18	12.360782			18	3175.3519
		19	12.357869			19	3174.7746
		20	12.353979			20	**3174**.1890
4	24	16	24.056877	15	2564	16	4858.1937
		17	24.053495			17	4856.4186
		18	24.051431			18	4855.1064
		19	24.048769			19	4854.3872
		20	24.047205			20	**4853**.7561
5	40	16	44.981014	16	4320	16	7332.7695
		17	44.976437			17	7329.8545
		18	44.973846			18	7325.5713
		19	44.971353			19	7322.5461
		20	44.970252			20	**7320**.1068
6	72	16	78.187644	17	5346	16	11014.169
		17	78.173268			17	11004.299
		18	78.163358			18	10994.873
		19	78.151981			19	10984.895
		20	78.143569			20	**10978**.622
7	126	16	134.26988	18	7398	16	16469.091
		17	134.21522			17	16445.457
		18	134.17305			18	16431.764
		19	134.13115			19	16418.296
		20	134.10709			20	**16406**.358
9	306	16	363.67296	19	10668	16	24575.872
		17	363.59590			17	24516.534
		18	363.50742			18	24463.542
		19	363.41738			19	24443.476
		20	363.34567			20	**24417**.472
10	500	16	553.82278	20	17400	16	36402.676
		17	553.57125			17	36296.753
		18	553.38179			18	36250.908
		19	553.21188			19	36218.806
		20	553.05527			20	**36195**.348
11	582	16	869.23401	21	27720	16	53878.723
		17	868.82650			17	53724.682
		18	868.45366			18	53647.201
		19	868.15131			19	53567.621
		20	**868**.01070			20	**53524**.085

Table 1 continued

n	Lower bound	d	Upper bound	n	Lower bound	d	Upper bound
12	840	16	1356.5778	22	49896	16	81376.460
		17	1356.1536			17	81085.186
		18	1355.8837			18	80962.164
		19	1355.4776			19	80860.092
		20	**1355**.2976			20	**80810**.158
13	1154	16	2066.3465	23	93150	16	123328.40
		17	2065.5348			17	122796.10
		18	2064.9493			18	122657.49
		19	2064.4859			19	122481.07
		20	**2064**.0029			20	**122351**.67

Dimension 8 is omitted since there the linear programming bound is sharp. New records after rounding down to the nearest integer are bolded. Lower bounds are taken from [5]

this verification procedure since our main goal here is to showcase the speed of our approach for this type of problems. Note that the bounds we report for $d = 16$ are slightly different from the bounds reported in [5] since their verification procedure increases the bounds by a configurable parameter $\varepsilon > 0$.

6.2 Binary sphere packing

The m-sphere packing problem asks for the optimal sphere packing density in Euclidean space using spheres of m prescribed sizes. For $m = 1$ this is the well-known sphere packing problem, for which the linear programming bound by Cohn and Elkies has been used to prove the optimality of the E_8 root lattice in $\mathbb{R}^8$ and the Leech lattice in $\mathbb{R}^{24}$ [6, 44, 45]. In [7], de Laat, Oliveira, and Vallentin generalize this bound to the m-sphere packing problem, and they use this to compute bounds for the binary sphere packing problem in dimensions $2, \dots, 5$ with radii $(r/1000, 1)$, $r = 200, \dots, 1000$. For smaller r and dimensions higher than 5 computations were prevented by numerical instabilities. Here we show this bound can be modeled as a univariate polynomial matrix program using matrices of size m. We use this to perform computations for a larger range of radii and higher dimensions, which allows us to make new qualitative observations about the behavior of these bounds.

Before stating the bound, we recall some definitions. A function $f\colon \mathbb{R}^n \to \mathbb{R}$ is a Schwarz function if it is infinitely differentiable, and if any derivative of $f(v)$ multiplied with any monomial in $v_1, \dots, v_n$ is a bounded function. If $f\colon \mathbb{R}^n \to \mathbb{R}$ is a radial function, then for $t \geq 0$ we write $f(t)$ for the common value of f on vectors v of norm t. A matrix-valued Schwartz function $f\colon \mathbb{R}^n \to \mathbb{R}^{m\times m}$ is a matrix-valued functions whose every component function is a Schwartz function. We define the Fourier transform of such functions entrywise:

$$\widehat{f}(v)_{rs} = \int f(w)_{rs} e^{-2\pi i \langle v,w\rangle}\, dw.$$

Theorem 2 [7, Theorem 5.1] *Let $R_1, \dots, R_m > 0$. Suppose $f\colon \mathbb{R}^n \to \mathbb{R}^{m\times m}$ is a radial, matrix-valued Schwartz function that satisfies the following:*

1. *The matrix $\widehat{f}(0) - W$ is positive semidefinite, where*

$$W_{rs} = (\operatorname{vol} B(R_r))^{1/2} (\operatorname{vol} B(R_s))^{1/2}$$

   and $B(R)$ is the ball of radius R.
2. *The matrix $\widehat{f}(t)$ is positive semidefinite for every $t > 0$.*
3. *$f_{rs}(t) \le 0$ whenever $t \ge R_r + R_s$, for $r, s = 1, \dots, m$.*

Then the density of any sphere packing of spheres of radii $R_1, \dots, R_m$ in the Euclidean space $\mathbb{R}^n$ is at most $\max\{f_{rr}(0) : r = 1, \dots, m\}$.

Following [7] we parametrize $\widehat{f}$ as

$$\widehat{f}(t) = \sum_{k=0}^{d} A^{(k)} t^{2k} e^{-\pi t^2},$$

for some symmetric matrices $A^{(0)}, \dots, A^{(d)}$. By [7, Lemma 5.2] we then have

$$f(t) = \sum_{k=0}^{d} A^{(k)} \frac{k!}{\pi^k} L_k^{n/2-1}(\pi t^2) e^{-\pi t^2},$$

where $L_k^{n/2-1}$ is the degree k Laguerre polynomial with parameter $n/2 - 1$. Because positive factors do not influence the positivity, the Gaussians can be ignored, resulting in the following polynomial matrix program (where some of the constraints use only degree 0 polynomials):

$$\begin{aligned}
\text{minimize} \quad & M \\
\text{subjectto} \quad & -W + A^{(0)} \succeq 0 \\
& \sum_{k=0}^{d} A^{(k)} x^k \succeq 0 \text{ on } \mathbb{R}_+ \\
& -\sum_{k=0}^{d} A_{rs}^{(k)} \frac{k!}{\pi^k} L_k^{n/2-1}(\pi x) \succeq 0 \text{ on } S(G_{rs}), \qquad && \text{for } 1 \le r \le s \le m \\
& M - \sum_{k=0}^{d} A_{rr}^{(k)} \frac{k!}{\pi^k} L_k^{n/2-1}(0) \succeq 0, && \text{for } 1 \le r \le m,
\end{aligned}$$

where we used the transformation $x = t^2$, and define $G_{rs} = \{x - (R_r + R_s)^2\}$.

The plots in [7] suggest that the binary sphere packing bounds become very bad as the ratio of the radii r tends to 0. Using the extra data we collect, we see the bounds are not convex in r and in fact the bounds seem to become very good as r tends to

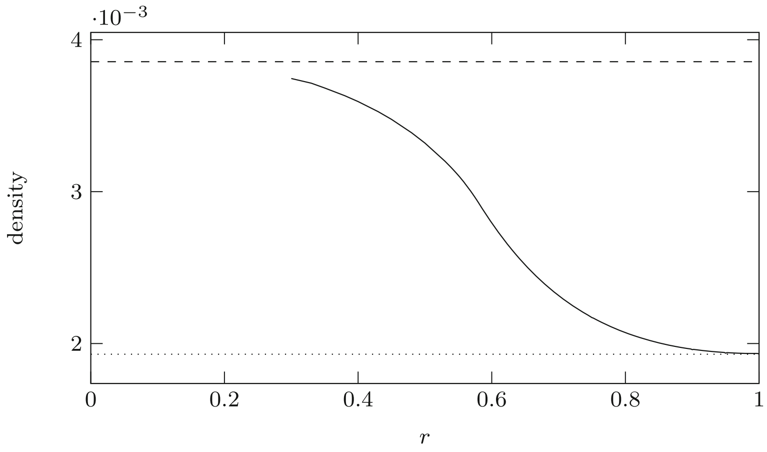

Fig. 3 The binary sphere packing bound in dimension 24. The dotted line is the maximum density of single sphere packings and the dashed line the optimal density when r tends to 0

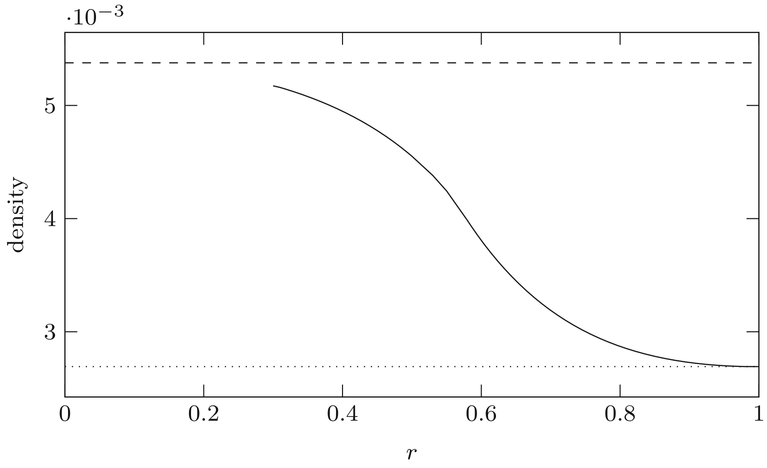

Fig. 4 The binary sphere packing bound in 23 dimensions. The horizontal axis represents the ratio between the radii of the small and the large sphere, and the vertical axis the bound

0. For dimension 24, the plot in Fig. 3 seems to suggest that as the ratio of the radii r tends to zero the binary sphere packing bound converges to $\Delta + (1 - \Delta)\Delta$, where Δ denotes the optimal sphere packing density. By [45, 46] this is the optimal limiting binary sphere packing density. As the computations for dimension 23 suggest (see Fig. 4), the bound more generally may go to $\delta + \delta(1 - \delta)$, where δ is the optimal value of the Cohn-Elkies linear programming bound.

In dimension 2, the best known upper bound for the binary sphere packing problem is due to Florian [47]. In [7], the binary sphere packing bound was calculated for $r \geq 0.2$, and in this regime, the bound is worse than Florian's bound. We compute the bound for $r \geq 0.035$, which is enough to show the bound improves on Florian's bound for small r; see Fig. 5.

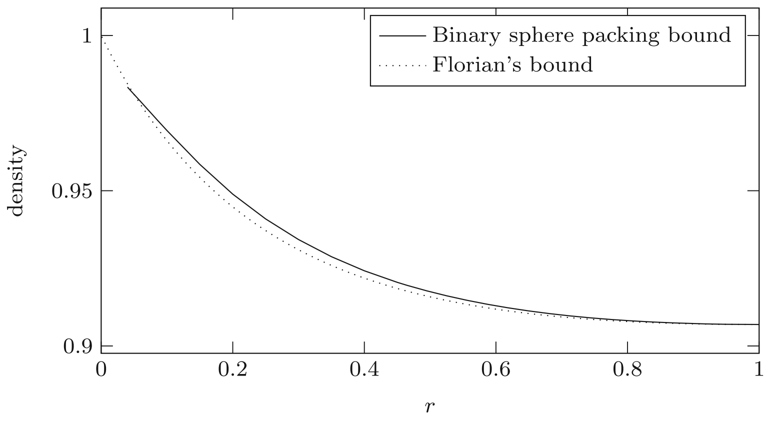

Fig. 5 The binary sphere packing bound in dimension $n = 2$ and Florian's bound

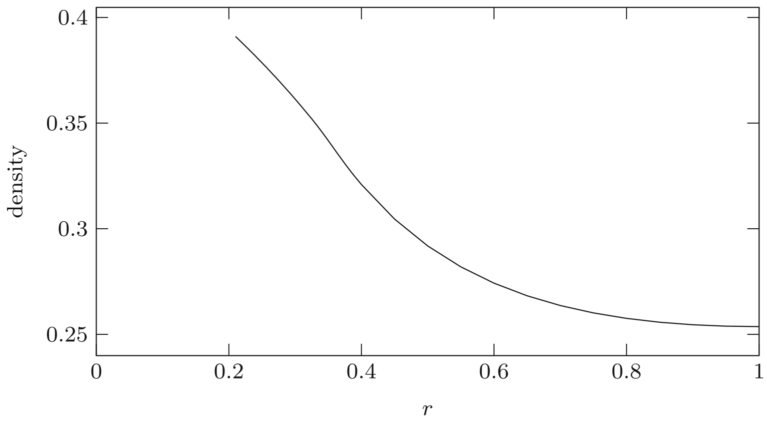

Fig. 6 Binary sphere packing upper bounds in dimension $n = 8$

Since the binary sphere packing density only depends on the ratio $r = R_1/R_2$ of the radii of the spheres, we may scale both radii by a constant factor $s > 0$ for the calculations. We observed that using scaled radii instead of $(r, 1)$ can lead to better numerical conditioning of the problem and to better bounds. The difference can be especially large for small r, and decreases when increasing r. For example, in dimension 2 with radii $(1/10, 1)$, the bound for degree $d = 31$ is 1.155 without scaling and 0.9697 with scaling factor $s = 1.35$.

In dimensions 8 and 24, we could compute the bound for $r \geq 0.15$ respectively $r \geq 0.3$, see Figs. 6 and 3. This required degrees 71; for small r we computed the bounds with degree $d = 91$ to make sure that increasing the degree would not change the plot visibly. We scaled the radii with $s = 19/10$. In Fig. 3, we added a dotted line indicating the optimal sphere packing density $\Delta_{24} = \pi^{12}/12!$ [45], and a dashed line indicating the optimal limiting density. In dimensions 2 and 8 we omitted these lines since the curve of the bound is less clear in those dimensions; in dimension 2 the bound is still convex, and in dimension 8 it is unclear how fast the bound flattens (Fig. 4).

Acknowledgements We thank Henry Cohn and Fernando Oliveira for their helpful comments. We also thank the anonymous reviewers whose suggestions helped improve the paper.

Funding The authors declare that no funds, grants, or other support were received during the preparation of this manuscript.

Data availability The data used to generate the table and the figures is available in the arXiv version of this paper.

Declarations

Conflict of interest The authors have no relevant financial or non-financial interests to disclose.

A Symmetric description of a symmetric semialgebraic set

In this appendix we generalize the argument from [5, Lemma 3.1] to arbitrary finite groups, to show that an invariant semialgebraic set is the semialgebraic set of invariant polynomials.

Let Γ be a finite group with a linear action on $\mathbb{R}^n$ and consider the linear action on $\mathbb{R}[x]$ by $\gamma p(x) = p(\gamma^{-1}x)$. Let G be a finite set of polynomials such that the semialgebraic set $\mathcal{S}(G) = \{x \in \mathbb{R}^n : g(x) \geq 0 \text{ for } g \in G\}$ is Γ-invariant. We will show $\mathcal{S}(G)$ is the semialgebraic set of finitely many invariant polynomials. Since $\mathcal{S}(G) = \bigcap_{g \in G} \mathcal{S}(\Gamma g)$, it is sufficient to show this for $\mathcal{S}(\Gamma g)$.

For $g \in G$ we define the Γ-invariant polynomials

$$g_k(x) = \sum_{\substack{\Gamma' \subseteq \Gamma \\ |\Gamma'| = k}} \prod_{\gamma \in \Gamma'} \gamma g(x)$$

for $k = 1, \dots, |\Gamma|$. Then,

$$\mathcal{S}(\Gamma g) \subseteq \mathcal{S}(\{g_1, \dots, g_{|\Gamma|}\})$$

since $\gamma g(x) \geq 0$ for all $\gamma \in \Gamma$ implies $g_k(x) \geq 0$ for all k.

For the reverse inclusion we suppose we have a point x with $g(x) < 0$ and $g_k(x) \geq 0$ for $k \geq 2$. We will argue that $g_1(x) < 0$.

Define

$$T_k = \sum_{\substack{\Gamma' \subseteq \Gamma \\ |\Gamma'|=k \\ e \notin \Gamma'}} \prod_{\gamma \in \Gamma'} \gamma g(x),$$

where we denote the identity element of Γ by e. Then $T_k \leq 0$ for $k = |\Gamma|$, since there are no subsets of Γ of size $|\Gamma|$ not containing the identity. If $T_k \leq 0$ for $k \geq 2$, then $g_k(x) = g(x)T_{k-1} + T_k \geq 0$ implies $g(x)T_{k-1} \geq 0$ and hence $T_{k-1} \leq 0$. By induction we then have $T_1 \leq 0$. Hence,

$$g_1(x) = \sum_{\gamma \in \Gamma} \gamma g(x) = g(x) + T_1 \leq g(x) < 0.$$

So $\mathcal{S}(\Gamma g) = \mathcal{S}(\{g_1, \ldots, g_{|\Gamma|}\})$.

B Nonexistence of an invariant minimal unisolvent set

In this appendix we give an example in which an invariant minimal unisolvent set does not exist even though this is not apparent from the dimensions of the spaces and the orbit sizes.

Let $\Gamma = D_4$ be the dihedral group $\langle s, d : d^4 = s^2 = e,\ ds = sd^{-1} \rangle$ with the action on $\mathbb{R}^2$ defined by

$$\theta(s)(x, y) = (y, x), \qquad \theta(d)(x, y) = (-y, x).$$

The invariant ring $\mathbb{R}[x, y]^\Gamma$ is generated by $\phi_1 = x^2 + y^2$ and $\phi_2 = x^2 y^2$. Take $2d = 6$. Then $\dim \mathbb{R}[x, y]^\Gamma_{\leq 2d} = 6$, and $\dim \mathbb{R}[x, y]_{\leq 2d} = \binom{2+6}{2} = 28$.

Let M be an invariant set of points with $|M| = \dim \mathbb{R}[x, y]_{\leq 2d} = 28$, and suppose M is unisolvent. We will show that this leads to a contradiction. By Lemma 2, the set R of representatives of the orbits is minimal unisolvent for $\mathbb{R}[x, y]^\Gamma_{\leq 2d}$, and thus has size $\dim \mathbb{R}[x, y]^\Gamma_{\leq 2d} = 6$.

The orbits of Γ acting on $\mathbb{R}^2$ have size 1 (the origin $(0, 0)$), size 4 (generated by points (x, y) with $|x| = |y|$, $x = 0$, or $y = 0$) and size 8 (generated by points (x, y) with $|x| \neq |y|$ and $x, y \neq 0$). Since there is only one orbit of odd size, and $|M| = \dim \mathbb{R}[x, y]_{\leq 2d}$ is even, all orbits of M have even size. From the equations

$$4k + 8l = |S| = 28 \quad \text{and} \quad k + l = |R| = 6$$

we know that there are $k = 5$ points in R corresponding to orbits of size 4, and $l = 1$ point corresponding to an orbit of size 8.

Let r be the norm of a point corresponding to the orbit of size 8. Note that $\|(x, y)\|^2 = \phi_1(x, y)$ is invariant under the action of Γ, hence all points in an orbit have the same norm. Define the polynomial

$$p(x, y) = xy(x + y)(x - y)(x^2 + y^2 - r^2).$$

Then $p(x, y) = 0$ for all $(x, y) \in M$, and $\deg p = 6 = 2d$ so $p \in \mathbb{R}[x, y]_{\leq 2d}$, but $p \neq 0$. Hence M is not unisolvent.

References

1. Delsarte, P., Goethals, J.M., Seidel, J.J.: Spherical codes and designs. Geom. Dedicata **6**(3), 363–388 (1977). https://doi.org/10.1007/BF03187604
2. Musin, O.R.: The kissing number in four dimensions. Ann. Math. (2) **168**(1), 1–32 (2008). https://doi.org/10.4007/annals.2008.168.1
3. Bachoc, C., Vallentin, F.: New upper bounds for kissing numbers from semidefinite programming. J. Am. Math. Soc. **21**(3), 909–924 (2008). https://doi.org/10.1090/S0894-0347-07-00589-9
4. Mittelmann, H.D., Vallentin, F.: High accuracy semidefinite programming bounds for kissing numbers. Exp. Math. **19**(2), 175–179 (2010). https://doi.org/10.1080/10586458.2010.10129070. arXiv:0902.1105
5. Machado, F.C., de Oliveira Filho, F.M.: Improving the semidefinite programming bound for the kissing number by exploiting polynomial symmetry. Exp. Math. **27**(3), 362–369 (2018). https://doi.org/10.1080/10586458.2017.1286273
6. Cohn, H., Elkies, N.: New upper bounds on sphere packings. I. Ann. Math. (2) **157**(2), 689–714 (2003). https://doi.org/10.4007/annals.2003.157.689
7. de Laat, D., de Oliveira Filho, F.M., Vallentin, F.: Upper bounds for packings of spheres of several radii. Forum Math. Sigma **2**, e23 (2014). https://doi.org/10.1017/fms.2014.24
8. Dostert, M., Guzmán, C., de Oliveira Filho, F.M., Vallentin, F.: New upper bounds for the density of translative packings of three-dimensional convex bodies with tetrahedral symmetry. Discrete Comput. Geom. **58**(2), 449–481 (2017). https://doi.org/10.1007/s00454-017-9882-y
9. Yudin, V.A.: Minimum potential energy of a point system of charges. Diskret. Mat. **4**(2), 115–121 (1992). https://doi.org/10.1515/dma.1993.3.1.75
10. Cohn, H., Woo, J.: Three-point bounds for energy minimization. J. Am. Math. Soc. **25**(4), 929–958 (2012). https://doi.org/10.1090/S0894-0347-2012-00737-1
11. de Laat, D.: Moment methods in energy minimization: New bounds for Riesz minimal energy problems. Trans. Am. Math. Soc. **373**(2), 1407–1453 (2019). https://doi.org/10.1090/tran/7976. arXiv:1610.04905
12. Chirre, A., Gonçalves, F., de Laat, D.: Pair correlation estimates for the zeros of the zeta function via semidefinite programming. Adv. Math. **361**, 106926 (2020). https://doi.org/10.1016/j.aim.2019.106926
13. Simmons-Duffin, D.: A semidefinite program solver for the conformal bootstrap. J. High Energy Phys. **2015**(6), 174 (2015). https://doi.org/10.1007/JHEP06(2015)174
14. Yamashita, M., Fujisawa, K., Nakata, K., Nakata, M., Fukuda, M., Kobayashi, K., Goto, K.: A high-performance software package for semidefinite programs: SDPA 7. Research Report B-460, Department of Mathematical and Computing Science, Tokyo Institute of Technology, Tokyo, Japan (2010)
15. Lofberg, J., Parrilo, P.: From coefficients to samples: a new approach to SOS optimization. In: 2004 43rd IEEE Conference on Decision and Control (CDC) (IEEE Cat. No.04CH37601), pp. 3154–3159. IEEE, Nassau (2004). https://doi.org/10.1109/CDC.2004.1428957
16. Liu, Z., Vandenberghe, L.: Low-rank structure in semidefinite programs derived from the KYP lemma. In: 2007 46th IEEE Conference on Decision and Control, pp. 5652–5659. IEEE, New Orleans, LA, USA (2007). https://doi.org/10.1109/CDC.2007.4434343
17. Benson, S.J.: DSDP5: software for semidefinite programming. ACM Trans. Math. Softw. 21 (2005)
18. Toh, K.C., Todd, M.J., Tütüncü, R.H.: On the implementation and usage of SDPT3—A Matlab software package for semidefinite-quadratic-linear programming, version 4.0. In: Anjos, M.F., Lasserre, J.B.

(eds.) Handbook on Semidefinite, Conic and Polynomial Optimization, vol. 166, pp. 715–754. Springer US, Boston (2012). https://doi.org/10.1007/978-1-4614-0769-0_25
19. Landry, W., Simmons-Duffin, D.: Scaling the semidefinite program solver SDPB. arXiv:1909.09745 [hep-th] (2019)
20. Bezanson, J., Edelman, A., Karpinski, S., Shah, V.B.: Julia: a fresh approach to numerical computing. SIAM Rev. **59**(1), 65–98 (2017). https://doi.org/10.1137/141000671
21. Johansson, F.: Arb: Efficient arbitrary-precision midpoint-radius interval arithmetic. IEEE Trans. Comput. **66**(8), 1281–1292 (2017). https://doi.org/10.1109/TC.2017.2690633
22. Helmberg, C., Rendl, F., Vanderbei, R.J., Wolkowicz, H.: An interior-point method for semidefinite programming. SIAM J. Optim. **6**(2), 342–361 (1996). https://doi.org/10.1137/0806020
23. Kojima, M., Shindoh, S., Hara, S.: Interior-point methods for the monotone semidefinite linear complementarity problem in symmetric matrices. SIAM J. Optim. **7**(1), 86–125 (1997). https://doi.org/10.1137/S1052623494269035
24. Monteiro, R.D.C.: Primal-dual path-following algorithms for semidefinite programming. SIAM J. Optim. **7**(3), 663–678 (1997). https://doi.org/10.1137/S1052623495293056
25. Mehrotra, S.: On the implementation of a primal-dual interior point method. SIAM J. Optim. **2**(4), 575–601 (1992). https://doi.org/10.1137/0802028
26. Toh, K.C.: A note on the calculation of step-lengths in interior-point methods for semidefinite programming. Comput. Optim. Appl. **21**(3), 301–310 (2002). https://doi.org/10.1023/A:1013777203597
27. Putinar, M.: Positive polynomials on compact semi-algebraic sets. Indiana Univ. Math. J. **42**(3), 969–984 (1993)
28. Scherer, C.W., Hol, C.W.J.: Matrix sum-of-squares relaxations for robust semi-definite programs. Math. Program. **107**(1), 189–211 (2006). https://doi.org/10.1007/s10107-005-0684-2
29. Klep, I., Schweighofer, M.: Pure states, positive matrix polynomials and sums of Hermitian squares. Indiana Univ. Math. J. **59**(3), 857–874 (2010). https://doi.org/10.1512/iumj.2010.59.4107. arXiv:0907.2260
30. Gatermann, K., Parrilo, P.A.: Symmetry groups, semidefinite programs, and sums of squares. J. Pure Appl. Algebra **192**(1), 95–128 (2004). https://doi.org/10.1016/j.jpaa.2003.12.011
31. Serre, J.P.: Linear Representations of Finite Groups, corr. 5th print edn. No. 42 in Graduate Texts in Mathematics. Springer, New York (1996)
32. Humphreys, J.E.: Reflection Groups and Coxeter Groups. Cambridge University Press, Cambridge (1990). https://doi.org/10.1017/cbo9780511623646
33. Chung, K.C., Yao, T.H.: On lattices admitting unique Lagrange interpolations. SIAM J. Numer. Anal. **14**(4), 735–743 (1977). https://doi.org/10.1137/0714050
34. Sommariva, A., Vianello, M.: Computing approximate Fekete points by QR factorizations of Vandermonde matrices. Comput. Math. Appl. **57**(8), 1324–1336 (2009). https://doi.org/10.1016/j.camwa.2008.11.011
35. Taylor, M.A., Wingate, B.A., Vincent, R.E.: An algorithm for computing Fekete points in the triangle. SIAM J. Numer. Anal. **38**(5), 1707–1720 (2000). https://doi.org/10.1137/S0036142998337247
36. Fieker, C., Hart, W., Hofmann, T., Johansson, F.: Nemo/hecke: Computer algebra and number theory packages for the julia programming language. In: Proceedings of the 2017 ACM on International Symposium on Symbolic and Algebraic Computation, ISSAC ’17, pp. 157–164. ACM, New York, NY, USA (2017). https://doi.org/10.1145/3087604.3087611
37. Pfender, F., Ziegler, G.M.: Kissing numbers, sphere packings, and some unexpected proofs. Not. Am. Math. Soc. **51**(8), 873–883 (2004)
38. Dostert, M., de Laat, D., Moustrou, P.: Exact semidefinite programming bounds for packing problems. SIAM J. Optim. **31**(2), 1433–1458 (2021). https://doi.org/10.1137/20M1351692
39. Borchers, B.: CSDP, A C library for semidefinite programming. Optim. Methods Softw. **11**(1–4), 613–623 (1999). https://doi.org/10.1080/10556789908805765
40. Skajaa, A., Ye, Y.: A homogeneous interior-point algorithm for nonsymmetric convex conic optimization. Math. Program. **150**(2), 391–422 (2015). https://doi.org/10.1007/s10107-014-0773-1
41. Papp, D., Yıldız, S.: On “A Homogeneous Interior-Point Algorithm for Non-Symmetric Convex Conic Optimization”. arXiv:1712.00492 [math] (2018)
42. Papp, D., Yıldız, S.: Sum-of-squares optimization without semidefinite programming. arXiv:1712.01792 [math] (2018)
43. Johansson, F.: Faster arbitrary-precision dot product and matrix multiplication. arXiv:1901.04289 [cs] (2019)

44. Viazovska, M.: The sphere packing problem in dimension 8. Ann. Math. **185**(3), 991–1015 (2017). https://doi.org/10.4007/annals.2017.185.3.7
45. Cohn, H., Kumar, A., Miller, S.D., Radchenko, D., Viazovska, M.: The sphere packing problem in dimension 24. Ann. Math. **185**(3), 1017–1033 (2017). https://doi.org/10.4007/annals.2017.185.3.8. arXiv:1603.06518
46. de Laat, D.: Optimal densities of packings consisting of highly unequal objects. arXiv:1603.01094 [math] (2016)
47. Florian, A.: Ausfüllung der Ebene durch Kreise. Rendiconti del Circolo Matematico di Palermo **9**(3), 300–312 (1960). https://doi.org/10.1007/BF02851249