A specialized SDP solver for sums-of-squares problems in discrete geometry

Nando Leijenhorst
Joint work with David de Laat

Delft University of Technology, The Netherlands

January 18, 2022

The problem

$$\begin{array}{ll} \text{maximize} & \langle C, X\rangle + b^\mathsf{T} y \\ \text{subject to} & p_{X,y}(x) \geq 0 \quad \forall x \in S \end{array}$$

The problem

$$\begin{array}{ll} \text{maximize} & b^\mathsf{T} y \\ \text{subject to} & \sum_i y_i p_i(x) \geq 0 \quad -1 \leq x \leq 1 \end{array}$$

The problem

$$\text{maximize} \quad \langle C, X \rangle$$

$$\text{subject to} \quad \sum_i \langle X_i, Y_i(u,v,t) \rangle \geq 0 \quad \begin{pmatrix} 1 & u & v \\ u & 1 & t \\ v & t & 1 \end{pmatrix} \geq 0$$

The problem

$$\begin{aligned} \text{maximize} \quad & \langle C, X \rangle + b^{\mathsf{T}} y \\ \text{subject to} \quad & p_{X,y}(x) = \sum_i g_i(x) s_i(x) \end{aligned}$$

The problem

$$\begin{aligned} \text{maximize} \quad & \langle C, X \rangle + b^{\mathsf{T}} y \\ \text{subject to} \quad & p_{X,y}(x) = \sum_i g_i(x) \langle b(x) b(x)^{\mathsf{T}}, Y_i \rangle \end{aligned}$$

From polynomials to numbers

Consider

$$p(x) = \langle b(x)b(x)^T, Y\rangle.$$

We want:

$$\langle A_i, X\rangle = c_i \quad i = 1, \ldots, m$$

Two approaches:

- Coefficient matching
- Sampling

Coefficient matching

Let $\{w_i\}_{i=0}^{N}$ be a basis of polynomials up to degree d. Then

$$p(x) = \langle b(x)b(x)^T, Y \rangle$$

Coefficient matching

Let $\{w_i\}_{i=0}^N$ be a basis of polynomials up to degree d. Then

$$\sum_i p_i w_i(x) = p(x) = \langle b(x)b(x)^T, Y\rangle = \sum_i \langle A_i, Y\rangle w_i(x)$$

Coefficient matching

Let $\{w_i\}_{i=0}^N$ be a basis of polynomials up to degree d. Then

$$\sum_i p_i w_i(x) = p(x) = \langle b(x)b(x)^T, Y\rangle = \sum_i \langle A_i, Y\rangle w_i(x)$$

Coefficient matching

Let $\{w_i\}_{i=0}^N$ be a basis of polynomials up to degree d. Then

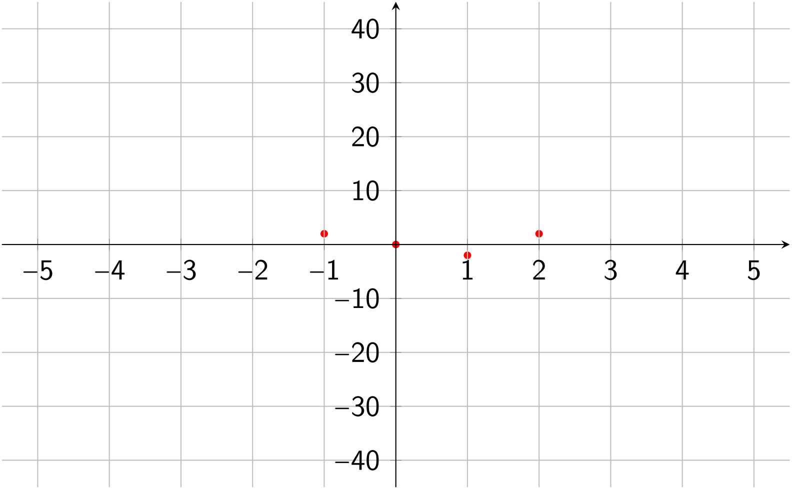
$$\sum_i p_i w_i(x) = p(x) = \langle b(x)b(x)^T, Y\rangle = \sum_i \langle A_i, Y\rangle w_i(x)$$

$$\iff$$

$$p_i = \langle A_i, Y\rangle, \quad i = 0, \ldots, N$$

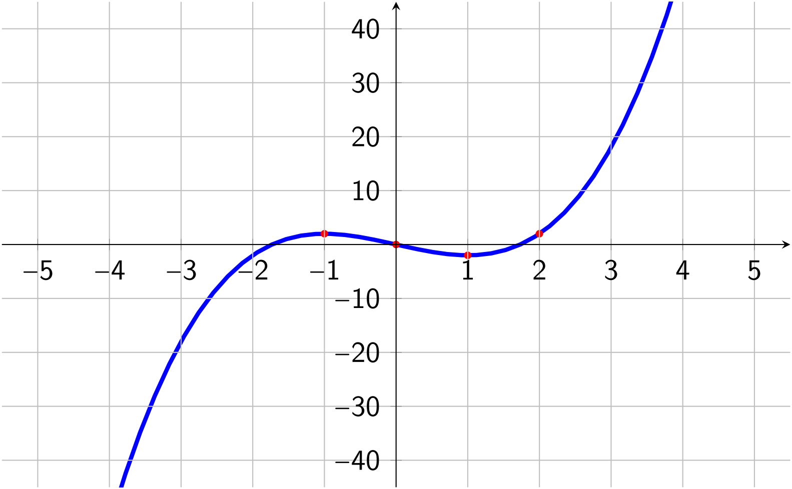
Sampling

$d + 1$ distinct points in $\mathbb{R}$ uniquely define a degree d polynomial:



Sampling

$d + 1$ distinct points in $\mathbb{R}$ uniquely define a degree d polynomial:



Sampling

A set $S = \{x_1, \dots, x_N\} \subset \mathbb{R}^n$ is unisolvent for n-variate polynomials of degree d if

$$p(x_i) = 0, \quad i = 1, \dots, N \iff p = 0.$$

Sampling

A set $S = \{x_1, \ldots, x_N\} \subset \mathbb{R}^n$ is unisolvent for n-variate polynomials of degree d if

$$p(x_i) = 0, \quad i = 1, \ldots, N \iff p = 0.$$

Thus $p(x) = \langle b(x)b(x)^T, Y \rangle$ if and only if $p(x_i) = \langle b(x_i)b(x_i)^T, Y \rangle$ for $i = 1, \ldots, N$.

Sampling

A set $S = \{x_1, \dots, x_N\} \subset \mathbb{R}^n$ is unisolvent for n-variate polynomials of degree d if

$$p(x_i) = 0, \quad i = 1, \dots, N \iff p = 0.$$

Thus $p(x) = \langle b(x)b(x)^T, Y\rangle$ if and only if $p(x_i) = \langle b(x_i)b(x_i)^T, Y\rangle$ for $i = 1, \dots, N$.
Note: Sampling is coefficient matching in the Lagrange basis $\{L_i\}$ corresponding to S, which are the polynomials of degree d with

$$L_i(x_j) = \delta_{ij}$$

Sampling versus coefficient matching

General coefficient matching:

$$p_i = \langle B_i, Y_i \rangle.$$

Sampling:

$$p(x_i) = \langle b(x_i) b(x_i)^T, Y \rangle.$$

Sampling versus coefficient matching

General coefficient matching: Possibly sparsity

$$p_i = \langle B_i, Y_i \rangle.$$

Sampling: Low-rank structure

$$p(x_i) = \langle b(x_i) b(x_i)^T, Y \rangle.$$

Clustered low-rank semidefinite program

$$\begin{aligned} \text{maximize} \quad & \sum_{j=1}^{J} \langle C^j, Y^j \rangle + \langle b, y \rangle \\ \text{subject to} \quad & \left\langle A_p^j, Y^j \right\rangle + B_p^j y = c_p^j, \quad j = 1, \ldots, J, p = 1, \ldots, N_J \\ & Y^j \succeq 0, \qquad\qquad\qquad\qquad j = 1, \ldots, J, \end{aligned}$$

with

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

and $A_p^j(l; r, s)$ of low rank.

Our contribution

A solver which supports:

Our contribution

A solver which supports:

- Primal-dual (SDPA, SDPT3, CSDP, SDPB, . . .)

Our contribution

A solver which supports:

- Primal-dual (SDPA, SDPT3, CSDP, SDPB, ...)
- Exploit low-rank structures (DSDP, SDPT3, SDPB (only rank 1))

Our contribution

A solver which supports:

- Primal-dual (SDPA, SDPT3, CSDP, SDPB, . . .)
- Exploit low-rank structures (DSDP, SDPT3, SDPB (only rank 1))
- Exploit clusters (SDPB (2 PSD matrices per cluster))

Our contribution

A solver which supports:

- Primal-dual (SDPA, SDPT3, CSDP, SDPB, ...)
- Exploit low-rank structures (DSDP, SDPT3, SDPB (only rank 1))
- Exploit clusters (SDPB (2 PSD matrices per cluster))
- High precision (SDPA-GMP/QD/DD, SDPB)

Our contribution

A solver which supports:

- Primal-dual (SDPA, SDPT3, CSDP, SDPB, . . .)
- Exploit low-rank structures (DSDP, SDPT3, SDPB (only rank 1))
- Exploit clusters (SDPB (2 PSD matrices per cluster))
- High precision (SDPA-GMP/QD/DD, SDPB)
- Parallel computations (SDPARA, SDPB, CSDP)

Our contribution

A solver which supports:

- Primal-dual (SDPA, SDPT3, CSDP, SDPB, . . .)
- Exploit low-rank structures (DSDP, SDPT3, SDPB (only rank 1))
- Exploit clusters (SDPB (2 PSD matrices per cluster))
- High precision (SDPA-GMP/QD/DD, SDPB)
- Parallel computations (SDPARA, SDPB, CSDP)

We further investigate:

- Combining samples and symmetry

Our contribution

A solver which supports:

- Primal-dual (SDPA, SDPT3, CSDP, SDPB, . . .)
- Exploit low-rank structures (DSDP, SDPT3, SDPB (only rank 1))
- Exploit clusters (SDPB (2 PSD matrices per cluster))
- High precision (SDPA-GMP/QD/DD, SDPB)
- Parallel computations (SDPARA, SDPB, CSDP)

We further investigate:

- Combining samples and symmetry
- Finding good bases and samples

Our contribution

A solver which supports:

- Primal-dual (SDPA, SDPT3, CSDP, SDPB, ...)
- Exploit low-rank structures (DSDP, SDPT3, SDPB (only rank 1))
- Exploit clusters (SDPB (2 PSD matrices per cluster))
- High precision (SDPA-GMP/QD/DD, SDPB)
- Parallel computations (SDPARA, SDPB, CSDP)

We further investigate:

- Combining samples and symmetry
- Finding good bases and samples
- Numerical experiments showing speed and stability

Semidefinite programming solver

Main steps:

- Calculate the so-called Schur complement matrix $S_{pq} = \langle A_p X^{-1} A_q, Y \rangle$
- Solve a system $Dz = d$, where S is a leading principal submatrix of D.
- Use z to determine the increments of the variables

Using the low-rank structure

$$S_{pq} = \mathrm{Tr}\left(A_p X^{-1} A_q Y\right)$$

Using the low-rank structure

$$
\begin{aligned}
S_{pq} &= \mathrm{Tr}\left(A_p X^{-1} A_q Y\right) \\
&= \mathrm{Tr}\left(a_p a_p^T X^{-1} a_q a_q^T Y\right)
\end{aligned}
$$

Using the low-rank structure

$$\begin{aligned} S_{pq} &= \mathrm{Tr}\left(A_p X^{-1} A_q Y\right) \\ &= \mathrm{Tr}\left(a_p a_p^T X^{-1} a_q a_q^T Y\right) \\ &= \mathrm{Tr}\left(a_p^T X^{-1} a_q a_q^T Y a_p\right) \end{aligned}$$

Using the low-rank structure

$$
\begin{aligned}
S_{pq} &= \mathrm{Tr}\left(A_p X^{-1} A_q Y\right) \\
&= \mathrm{Tr}\left(a_p a_p^T X^{-1} a_q a_q^T Y\right) \\
&= \mathrm{Tr}\left(a_p^T X^{-1} a_q a_q^T Y a_p\right) \\
&= (a_p^T X^{-1} a_q)(a_q^T Y a_p)
\end{aligned}
$$

Using the clustering

Recall that $A^j_p = 0$ if p is not contained in cluster j. Thus

$$S^{ij}_{pq} = \langle A^i_p X^{-1} A^j_q, Y \rangle = \begin{cases} 0 & i \neq j \\ \langle A^i_p X^{-1} A^i_q, Y \rangle & i = j \end{cases}$$

Then $Dz = d$ is given by

$$\begin{pmatrix} S^1 & 0 & \cdots & 0 & -B^1 \\ 0 & S^2 & \cdots & 0 & -B^2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & S^J & -B^J \\ (B^1)^\mathsf{T} & (B^2)^\mathsf{T} & \cdots & (B^J)^\mathsf{T} & 0 \end{pmatrix} z = d$$

Using the clustering

Let $S = LL^T$ be a Cholesky decomposition of S. The matrix D has the decomposition

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

Examples

- Three-point bound for the kissing number (Bachoc and Vallentin, 2007)

Examples

- Three-point bound for the kissing number (Bachoc and Vallentin, 2007)
  - Multivariate

Examples

- Three-point bound for the kissing number (Bachoc and Vallentin, 2007)
  - Multivariate
  - Symmetry

Examples

- Three-point bound for the kissing number (Bachoc and Vallentin, 2007)
  - Multivariate
  - Symmetry
  - Extensive previous computations (Bachoc and Vallentin, 2009),(Mittelman and Vallentin, 2010),(Machado and Oliviera, 2018)

Examples

- Three-point bound for the kissing number (Bachoc and Vallentin, 2007)
    - Multivariate
    - Symmetry
    - Extensive previous computations (Bachoc and Vallentin, 2009),(Mittelman and Vallentin, 2010),(Machado and Oliviera, 2018)
- Binary sphere packing (de Laat, Oliviera, and Vallentin, 2014)

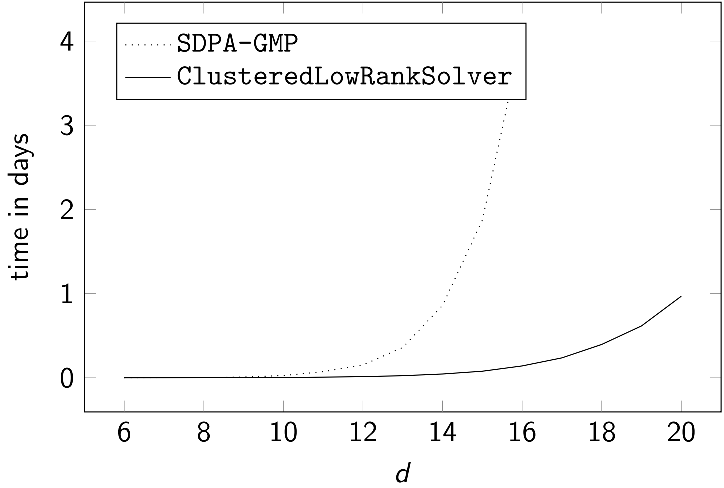
Examples

- Three-point bound for the kissing number (Bachoc and Vallentin, 2007)
  - Multivariate
  - Symmetry
  - Extensive previous computations (Bachoc and Vallentin, 2009),(Mittelman and Vallentin, 2010),(Machado and Oliviera, 2018)
- Binary sphere packing (de Laat, Oliviera, and Vallentin, 2014)
  - Polynomial <u>matrix</u> program

Examples

- Three-point bound for the kissing number (Bachoc and Vallentin, 2007)
  - Multivariate
  - Symmetry
  - Extensive previous computations (Bachoc and Vallentin, 2009),(Mittelman and Vallentin, 2010),(Machado and Oliviera, 2018)
- Binary sphere packing (de Laat, Oliviera, and Vallentin, 2014)
  - Polynomial <u>matrix</u> program
  - Numerically difficult

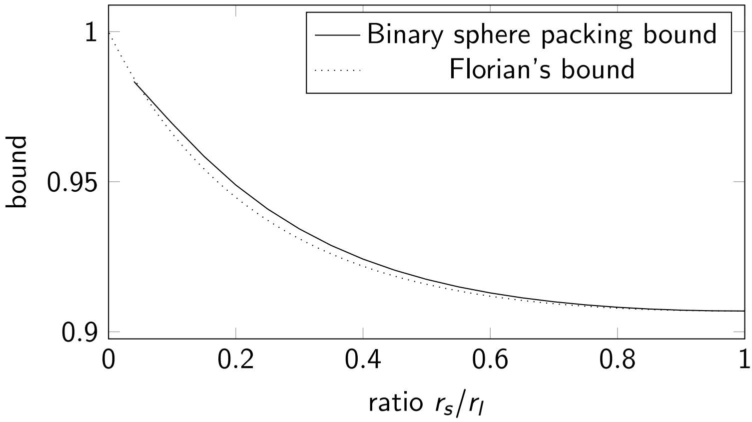
Results - Three-point bound for the kissing number



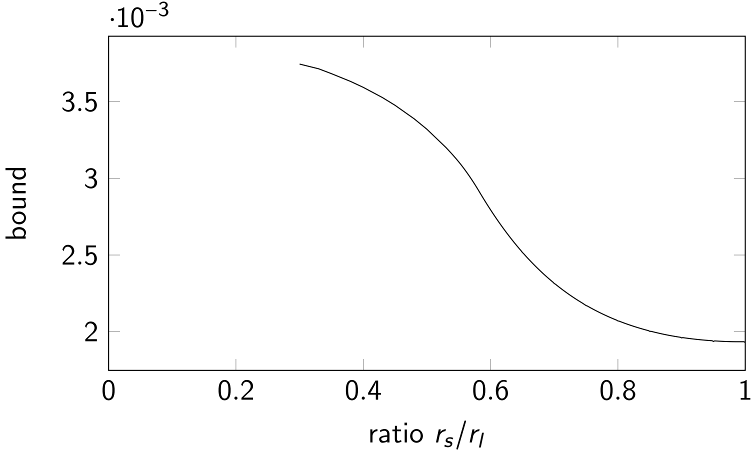
Results - Three-point bound for the kissing number

- 20× faster computations for previously computed bounds
- computations up to degree 20 (up to 16 before)
- new kissing number bounds in dimension $11 \leq n \leq 23$ and $25 \leq n \leq 48$

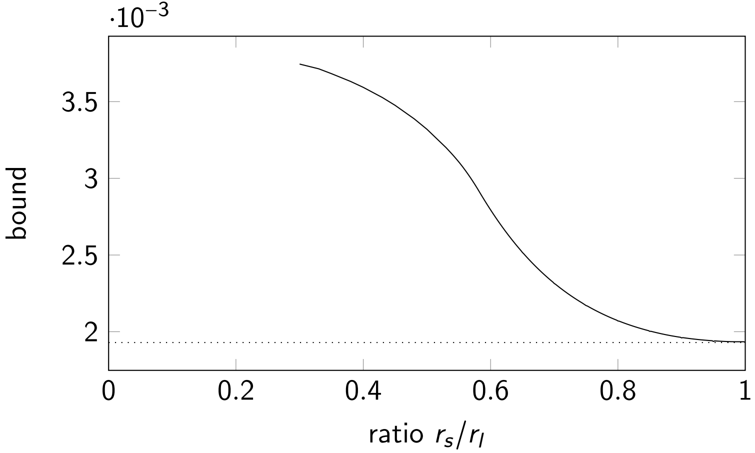
Results - Binary sphere packing (n=2)



Results - Binary sphere packing (n=24)

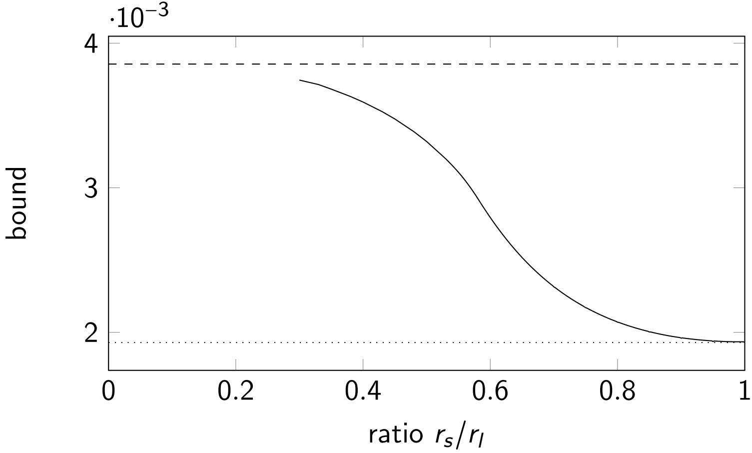


Results - Binary sphere packing (n=24)

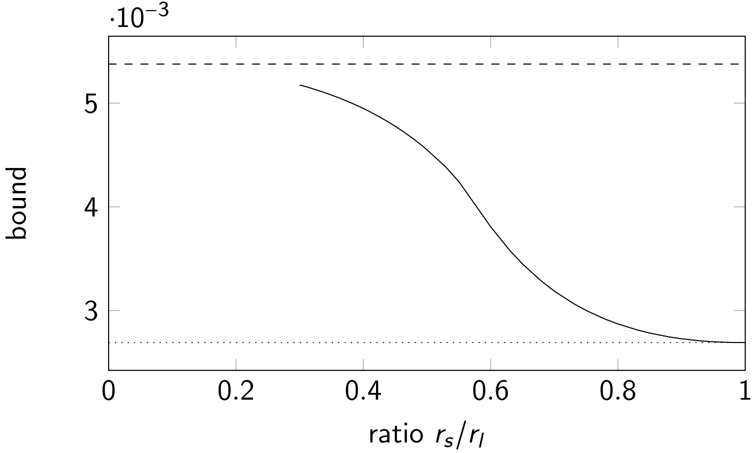


Optimal limiting density

Results - Binary sphere packing (n=24)



Results - Binary sphere packing (n=23)



Thank you!

Bonus slide - Iteratively improving samples and bases

Let $V = (p_i(x_j))_{ji}$ be the Vandermonde matrix of a basis $p = (p_1 \dots, p_N)$ with respect to the sample points $\{x_j\}$. Consider the QR decomposition $V = QR$.

Bonus slide - Iteratively improving samples and bases

Let $V = (p_i(x_j))_{ji}$ be the Vandermonde matrix of a basis $p = (p_1 \ldots, p_N)$ with respect to the sample points $\{x_j\}$. Consider the QR decomposition $V = QR$. Then Q is the Vandermonde matrix of a basis Rp with respect to the sample points $\{x_j\}$ and better conditioned than V.

Bonus slide - Iteratively improving samples and bases

Let $V = (p_i(x_j))_{ji}$ be the Vandermonde matrix of a basis $p = (p_1 \ldots, p_N)$ with respect to the sample points $\{x_j\}$. Consider the QR decomposition $V = QR$. Then Q is the Vandermonde matrix of a basis Rp with respect to the sample points $\{x_j\}$ and better conditioned than V.

Let $PV^T = QR$ be a pivoted QR decomposition. Let S be the set of samples corresponding to the first N pivots. Then $V|_S$ is relatively well-conditioned.