Exact semidefinite programming bounds for packing problems

Philippe Moustrou, UiT - The Arctic University of Norway
Joint work with M. Dostert (KTH) and D. de Laat (TU Delft).

Combinatorics and Geometric Days III - December 4, 2020

Contents

Exact semidefinite programming bounds for packing problems

Contents

Exact semidefinite programming bounds for packing problems

- Packing problems: What kind of problems?

Contents

Exact semidefinite programming bounds for packing problems

- Packing problems: What kind of problems?
- Semidefinite programming bounds: Optimization in the service of geometry.

Contents

Exact semidefinite programming bounds for packing problems

- Packing problems: What kind of problems?
- Semidefinite programming bounds: Optimization in the service of geometry.
- Exact: Why do we want exact bounds?

Contents

Exact semidefinite programming bounds for packing problems

- Packing problems: What kind of problems?
- Semidefinite programming bounds: Optimization in the service of geometry.
- Exact: Why do we want exact bounds?

Problem:

Usually semidefinite programming provides approximate numerical bounds.

Contents

Exact semidefinite programming bounds for packing problems

- Packing problems: What kind of problems?
- Semidefinite programming bounds: Optimization in the service of geometry.
- Exact: Why do we want exact bounds?

Problem:

Usually semidefinite programming provides approximate numerical bounds.

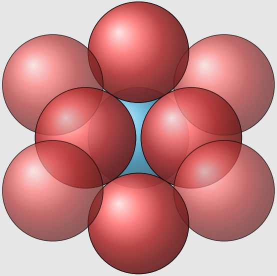
How can we turn these bounds into exact bounds?

Motivation: the kissing number problem

How many unit spheres can simultaneously touch a central unit sphere without overlapping?

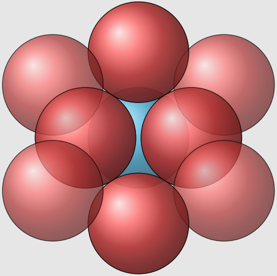
Motivation: the kissing number problem

How many unit spheres can simultaneously touch a central unit sphere without overlapping?



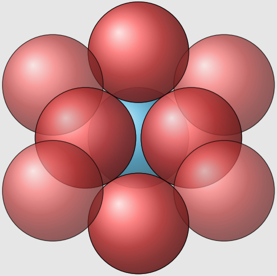
Motivation: the kissing number problem

How many unit spheres can simultaneously touch a central unit sphere without overlapping?



Motivation: the kissing number problem

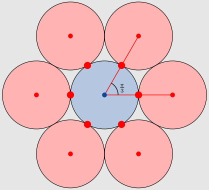
How many unit spheres can simultaneously touch a central unit sphere without overlapping?



Known in dimensions 1, 2, 3 (Schutte, vander Waerden, 1953),

4 (Musin, 2008), 8 and 24 (Levenshtein / Odlyzko, Sloane, 1979).

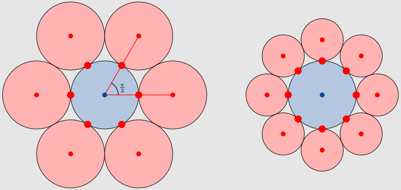
Formulation and generalizations



Kissing number:

$$\max\{|C|, \quad C \subset S^{n-1}, \quad x \cdot y \leq 1/2 \text{ for all } x \neq y \in C\}$$

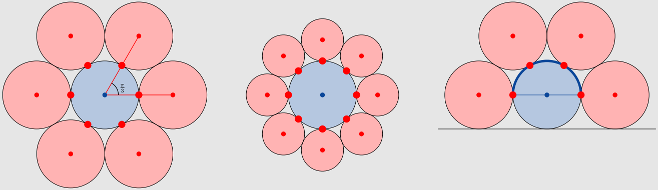
Formulation and generalizations



Spherical codes:

$$\max\{|C|, \quad C \subset S^{n-1}, \quad x \cdot y \leq \cos\theta \text{ for all } x \neq y \in C\}$$

Formulation and generalizations



One-sided kissing number (Musin, 2006):

$$\max\{|C|, \quad C \subset \mathrm{H}^{n-1}, \quad x \cdot y \leq 1/2 \text{ for all } x \neq y \in C\}$$

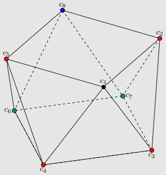
Goal and results

We are interested in special rigid structures, like:

Goal and results

We are interested in special rigid structures, like:

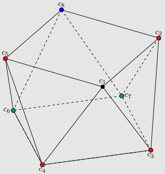
- The square antiprism, the unique optimal θ-spherical code in dimension 3 with $\cos\theta = (2\sqrt{2}-1)/7$ (Schütte-van der Waerden 1951, Danzer 1986).



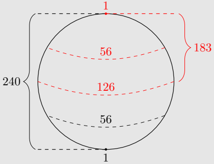
Goal and results

We are interested in special rigid structures, like:

- The square antiprism, the unique optimal θ-spherical code in dimension 3 with $\cos\theta = (2\sqrt{2}-1)/7$ (Schütte-van der Waerden 1951, Danzer 1986).



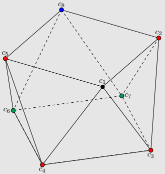
- For the Hemisphere in dimension 8: the E_8 lattice provides an optimal configuration (Bachoc-Vallentin, 2008). What about uniqueness?



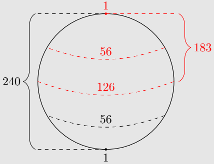
Goal and results

We are interested in special rigid structures, like:

- The square antiprism, the unique optimal θ-spherical code in dimension 3 with $\cos\theta = (2\sqrt{2}-1)/7$ (Schütte-van der Waerden 1951, Danzer 1986).



- For the Hemisphere in dimension 8: the E_8 lattice provides an optimal configuration (Bachoc-Vallentin, 2008). What about uniqueness?



- [Dostert, De Laat, M., 2020]: A general framework to prove optimality and uniqueness of such configurations.

These are optimization problems!

Let $G = (V, E)$ be the graph where:

These are optimization problems!

Let $G = (V, E)$ be the graph where:

- $V = S^{n-1}$ (or H^{n-1}),

These are optimization problems!

Let $G = (V, E)$ be the graph where:

- $V = S^{n-1}$ (or H^{n-1}),
- $\{x, y\} \in E$ if $x \cdot y > \cos\theta$.

These are optimization problems!

Let $G = (V, E)$ be the graph where:

- $V = S^{n-1}$ (or H^{n-1}),
- $\{x, y\} \in E$ if $x \cdot y > \cos\theta$.

Our problems boil down to computing the independence number of these graphs!

These are optimization problems!

Let $G = (V, E)$ be the graph where:

- $V = S^{n-1}$ (or H^{n-1}),
- $\{x, y\} \in E$ if $x \cdot y > \cos\theta$.

Our problems boil down to computing the independence number of these graphs!

- Lower bounds: Constructions.

These are optimization problems!

Let $G = (V, E)$ be the graph where:

- $V = S^{n-1}$ (or H^{n-1}),
- $\{x, y\} \in E$ if $x \cdot y > \cos\theta$.

Our problems boil down to computing the independence number of these graphs!

- Lower bounds: Constructions.
- Upper bounds:

These are optimization problems!

Let $G = (V, E)$ be the graph where:

- $V = S^{n-1}$ (or H^{n-1}),
- $\{x, y\} \in E$ if $x \cdot y > \cos\theta$.

Our problems boil down to computing the independence number of these graphs!

- Lower bounds: Constructions.
- Upper bounds:
  - For finite graphs: hierarchies of semidefinite upper bounds. (Lovász-Schrijver 1991, Lasserre 2001, Laurent 2007)

These are optimization problems!

Let $G = (V, E)$ be the graph where:

- $V = S^{n-1}$ (or H^{n-1}),
- $\{x, y\} \in E$ if $x \cdot y > \cos\theta$.

Our problems boil down to computing the independence number of these graphs!

- Lower bounds: Constructions.
- Upper bounds:
  - For finite graphs: hierarchies of semidefinite upper bounds. (Lovász-Schrijver 1991, Lasserre 2001, Laurent 2007)
  - For infinite graphs: Generalization of Lasserre's hierarchy (de Laat-Vallentin 2015), related to the previous 2-point (Delsarte-Goethals-Seidel 1977) and 3-point bounds (Bachoc-Vallentin 2008).

2-point bound for spherical codes (Delsarte-Goethals-Seidel 1977)

Based on two ingredients, related to the symmetries of the sphere:

2-point bound for spherical codes (Delsarte-Goethals-Seidel 1977)

Based on two ingredients, related to the symmetries of the sphere:

- Up to symmetry, a couple x, y of points in a θ-spherical code is uniquely determined by

$$u = x \cdot y, \quad \text{with } \begin{cases} u = 1 & x = y \\ u \in [-1, \cos\theta] & x \neq y \end{cases}$$

2-point bound for spherical codes (Delsarte-Goethals-Seidel 1977)

Based on two ingredients, related to the symmetries of the sphere:

- Up to symmetry, a couple x, y of points in a θ-spherical code is uniquely determined by

$$u = x \cdot y, \quad \text{with} \begin{cases} u = 1 & x = y \\ u \in [-1, \cos\theta] & x \neq y \end{cases}$$

- The normalized Gegenbauer polynomials $P_k^n(u)$ (with $P_k^n(1) = 1$), satisfying:

$$\text{For every } X \subset S^{n-1} \text{ finite, } \sum_{x,y \in X} P_k^n(x \cdot y) \geq 0.$$

2-point bound for spherical codes (Delsarte-Goethals-Seidel 1977)

Assume we have a polynomial f such that

- there exists coefficients $\alpha_0, \dots, \alpha_d \geq 0$ such that

$$f(u) = \sum_{k=0}^{d} \alpha_k P_k^n(u).$$

- $f(u) \leq -1$ for all $u \in [-1, \cos\theta]$

2-point bound for spherical codes (Delsarte-Goethals-Seidel 1977)

Assume we have a polynomial f such that

- there exists coefficients $\alpha_0, \ldots, \alpha_d \geq 0$ such that

$$f(u) = \sum_{k=0}^{d} \alpha_k P_k^n(u).$$

- $f(u) \leq -1$ for all $u \in [-1, \cos\theta]$

Then, if C is a θ-spherical code,

2-point bound for spherical codes (Delsarte-Goethals-Seidel 1977)

Assume we have a polynomial f such that

- there exists coefficients $\alpha_0, \ldots, \alpha_d \geq 0$ such that

$$f(u) = \sum_{k=0}^{d} \alpha_k P_k^n(u).$$

- $f(u) \leq -1$ for all $u \in [-1, \cos\theta]$

Then, if C is a θ-spherical code,

$$\sum_{x,y \in C} f(x \cdot y)$$

2-point bound for spherical codes (Delsarte-Goethals-Seidel 1977)

Assume we have a polynomial f such that

- there exists coefficients $\alpha_0, \ldots, \alpha_d \geq 0$ such that

$$f(u) = \sum_{k=0}^{d} \alpha_k P_k^n(u).$$

- $f(u) \leq -1$ for all $u \in [-1, \cos\theta]$

Then, if C is a θ-spherical code,

$$\sum_{k=0}^{d} \alpha_k \Big(\sum_{x,y \in C} P_k^n(x \cdot y) \Big) = \sum_{x,y \in C} f(x \cdot y)$$

2-point bound for spherical codes (Delsarte-Goethals-Seidel 1977)

Assume we have a polynomial f such that

- there exists coefficients $\alpha_0, \ldots, \alpha_d \geq 0$ such that
$$f(u) = \sum_{k=0}^{d} \alpha_k P_k^n(u).$$
- $f(u) \leq -1$ for all $u \in [-1, \cos\theta]$

Then, if C is a θ-spherical code,

$$0 \leq \sum_{k=0}^{d} \alpha_k \Big(\sum_{x,y \in C} P_k^n(x \cdot y) \Big) = \sum_{x,y \in C} f(x \cdot y)$$

2-point bound for spherical codes (Delsarte-Goethals-Seidel 1977)

Assume we have a polynomial f such that

- there exists coefficients $\alpha_0, \dots, \alpha_d \geq 0$ such that
$$f(u) = \sum_{k=0}^{d} \alpha_k P_k^n(u).$$
- $f(u) \leq -1$ for all $u \in [-1, \cos\theta]$

Then, if C is a θ-spherical code,

$$0 \leq \sum_{k=0}^{d} \alpha_k \Big(\sum_{x,y \in C} P_k^n(x \cdot y) \Big) = \sum_{x,y \in C} f(x \cdot y) \leq |C| f(1) + \sum_{x \neq y} f(x \cdot y)$$

2-point bound for spherical codes (Delsarte-Goethals-Seidel 1977)

Assume we have a polynomial f such that

- there exists coefficients $\alpha_0, \dots, \alpha_d \geq 0$ such that
$$f(u) = \sum_{k=0}^{d} \alpha_k P_k^n(u).$$
- $f(u) \leq -1$ for all $u \in [-1, \cos\theta]$

Then, if C is a θ-spherical code,

$$0 \leq \sum_{k=0}^{d} \alpha_k \Big(\sum_{x,y \in C} P_k^n(x \cdot y) \Big) = \sum_{x,y \in C} f(x \cdot y) \leq |C| f(1) + \sum_{x \neq y} f(x \cdot y) = |C|(f(1) - |C| + 1)$$

2-point bound for spherical codes (Delsarte-Goethals-Seidel 1977)

Assume we have a polynomial f such that

- there exists coefficients $\alpha_0, \ldots, \alpha_d \geq 0$ such that
$$f(u) = \sum_{k=0}^{d} \alpha_k P_k^n(u).$$
- $f(u) \leq -1$ for all $u \in [-1, \cos\theta]$

Then, if C is a θ-spherical code,

$$0 \leq \sum_{k=0}^{d} \alpha_k \Big(\sum_{x,y \in C} P_k^n(x \cdot y) \Big) = \sum_{x,y \in C} f(x \cdot y) \leq |C| f(1) + \sum_{x \neq y} f(x \cdot y) = |C|(f(1) - |C| + 1)$$

So

$$|C| \leq f(1) + 1$$

2-point bound for spherical codes (Delsarte-Goethals-Seidel 1977)

So for every $d \geq 0$, the size of a θ-spherical code is at most

$$\begin{aligned}\min\{M \in \mathbb{R} : \alpha_0, \ldots, \alpha_d &\geq 0,\\ f(1) &\leq M - 1,\\ f(u) &\leq -1 \text{ for all } u \in [-1, \cos\theta]\}\end{aligned}$$

where

$$f(u) = \sum_{k=0}^{d} \alpha_k P_k^n(u).$$

2-point bound for spherical codes (Delsarte-Goethals-Seidel 1977)

So for every $d \geq 0$, the size of a θ-spherical code is at most

$$\begin{aligned} \min\{M \in \mathbb{R} : \alpha_0, \ldots, \alpha_d &\geq 0, \\ f(1) &\leq M - 1, \\ f(u) &\leq -1 \text{ for all } u \in [-1, \cos\theta]\} \end{aligned}$$

where

$$f(u) = \sum_{k=0}^{d} \alpha_k P_k^n(u).$$

This is a linear programming bound.

3-point bound for spherical codes (Bachoc-Vallentin 2008)

Based on two ingredients related to the symmetries of the sphere:

3-point bound for spherical codes (Bachoc-Vallentin 2008)

Based on two ingredients related to the symmetries of the sphere:

- Up to symmetry, a triple of points x, y, z in a θ-spherical code is uniquely determined by

$$u = x \cdot y, \quad v = x \cdot z, \quad t = y \cdot z,$$

with (u, v, t) in

$$\begin{cases} \{(1,1,1)\} & x = y = z \\ \Delta_0 = \{(u,u,1) : u \in [-1, \cos\theta]\} & x \neq y = z \\ \Delta & x, y, z \text{ distinct} \end{cases}$$

where

$$\Delta = \{(u,v,t) : u,v,t \in [-1,\cos\theta], 1 + 2uvt - u^2 - v^2 - t^2 \geq 0\}$$

3-point bound for spherical codes (Bachoc-Vallentin 2008)

Based on two ingredients related to the symmetries of the sphere:

- Up to symmetry, a triple of points x, y, z in a θ-spherical code is uniquely determined by

$$u = x \cdot y, \quad v = x \cdot z, \quad t = y \cdot z,$$

  with (u, v, t) in

$$\begin{cases} \{(1,1,1)\} & x = y = z \\ \Delta_0 = \{(u,u,1) : u \in [-1, \cos\theta]\} & x \neq y = z \\ \Delta & x, y, z \text{ distinct} \end{cases}$$

  where

$$\Delta = \{(u,v,t) : u,v,t \in [-1, \cos\theta], 1 + 2uvt - u^2 - v^2 - t^2 \geq 0\}$$

- Matrix polynomials $S_k^n(u, v, t)$ satisfying:

$$\text{For every } X \subset S^{n-1} \text{ finite, } \sum_{x,y,z \in X} S_k^n(x \cdot y, x \cdot z, y \cdot t) \succeq 0.$$

3-point bound for spherical codes (Bachoc-Vallentin 2008)

Then for every $d \geq 0$, the size of a θ-spherical code is at most

$$\begin{aligned}\min\{M \in \mathbb{R} : \alpha_k \geq 0, F_k \succeq 0\\ &\sum_{k=0}^{d} \alpha_k + F(1,1,1) \leq M - 1,\\ &\sum_{k=0}^{d} \alpha_k P_k^n(u) + 3F(u,u,1) \leq -1 \text{ for all } u \in [-1, \cos\theta],\\ &F(u,v,t) \leq 0 \text{ for all } (u,v,t) \in \Delta\}\end{aligned}$$

where

$$F(u,v,t) = \sum_{k=0}^{d} \langle F_k, S_k^n(u,v,t)\rangle.$$

3-point bound for spherical codes (Bachoc-Vallentin 2008)

Then for every $d \geq 0$, the size of a θ-spherical code is at most

$$\min\{M \in \mathbb{R} : \alpha_k \geq 0, F_k \succeq 0$$

$$\sum_{k=0}^{d} \alpha_k + F(1,1,1) \leq M - 1,$$

$$\sum_{k=0}^{d} \alpha_k P_k^n(u) + 3F(u,u,1) \leq -1 \text{ for all } u \in [-1, \cos\theta],$$

$$F(u,v,t) \leq 0 \text{ for all } (u,v,t) \in \Delta\}$$

where

$$F(u,v,t) = \sum_{k=0}^{d} \langle F_k, S_k^n(u,v,t) \rangle.$$

This leads to semidefinite upper bounds using sums of squares.

Why exact bounds?

Assume we know a configuration C with $|C| = N$.

Why exact bounds?

Assume we know a configuration C with $|C| = N$.

- Any upper bound $< N + 1$ is enough to prove that C is optimal.

Why exact bounds?

Assume we know a configuration C with $|C| = N$.

- Any upper bound $< N + 1$ is enough to prove that C is optimal.
- Even if we do not solve the SDP exactly, if the numerical output of the solver is very close to N, it is not hard to prove a rigorous upper bound of the form $N + \varepsilon$.

Why exact bounds?

Assume we know a configuration C with $|C| = N$.

- Any upper bound $< N + 1$ is enough to prove that C is optimal.
- Even if we do not solve the SDP exactly, if the numerical output of the solver is very close to N, it is not hard to prove a rigorous upper bound of the form $N + \varepsilon$.

So why do we want an exact sharp bound?

Why exact bounds?

Assume we know a configuration C with $|C| = N$.

- Any upper bound $< N + 1$ is enough to prove that C is optimal.
- Even if we do not solve the SDP exactly, if the numerical output of the solver is very close to N, it is not hard to prove a rigorous upper bound of the form $N + \varepsilon$.

So why do we want an exact sharp bound?

- Optimization: When does a bound give the independence number?

Why exact bounds?

Assume we know a configuration C with $|C| = N$.

- Any upper bound $< N + 1$ is enough to prove that C is optimal.
- Even if we do not solve the SDP exactly, if the numerical output of the solver is very close to N, it is not hard to prove a rigorous upper bound of the form $N + \varepsilon$.

So why do we want an exact sharp bound?

- Optimization: When does a bound give the independence number?
- Geometry: Sharp bounds provide additional information on optimal configurations, leading to uniqueness proofs.

Recap

- For spherical codes, including kissing number:
  - 2-point bound $\rightarrow$ linear programming bound
  - 3-point bound $\rightarrow$ semidefinite programming bound

Recap

- For spherical codes, including kissing number:
  - 2-point bound $\rightarrow$ linear programming bound
  - 3-point bound $\rightarrow$ semidefinite programming bound

- For spherical codes in spherical caps, like hemisphere:
  - Delsarte bound does not apply anymore due to the lack of symmetry.
  - The 3-point bound can be adapted to a 2-point semidefinite programming bound (Bachoc-Vallentin 2009).

Results

Many examples of exact sharp LP bounds ...

Results

Many examples of exact sharp LP bounds ...

But very few cases in which SDP bound is proven to be sharp while LP is not:

Results

Many examples of exact sharp LP bounds ...

But very few cases in which SDP bound is proven to be sharp while LP is not:

- The Petersen code is the unique optimal $1/6$-code in dimension 4 (Bachoc-Vallentin 2009, Dostert-de Laat-M 2020).

Results

Many examples of exact sharp LP bounds ...

But very few cases in which SDP bound is proven to be sharp while LP is not:

- The Petersen code is the unique optimal $1/6$-code in dimension 4 (Bachoc-Vallentin 2009, Dostert-de Laat-M 2020).
- Numerically sharp for the square antiprism (Bachoc-Vallentin 2009) $\rightarrow$ Rigorous proof (Dostert-de Laat-M 2020)

Results

Many examples of exact sharp LP bounds ...

But very few cases in which SDP bound is proven to be sharp while LP is not:

- The Petersen code is the unique optimal 1/6-code in dimension 4 (Bachoc-Vallentin 2009, Dostert-de Laat-M 2020).
- Numerically sharp for the square antiprism (Bachoc-Vallentin 2009)
  $\rightarrow$ Rigorous proof (Dostert-de Laat-M 2020)
- E_8 gives an optimal configuration on the hemisphere in dimension 8 (Bachoc-Vallentin 2009)
  $\rightarrow$ Uniqueness (Dostert-de Laat-M 2020)

Solving an SDP: Rage against the machine precision

Solving an SDP: Rage against the machine precision

A semidefinite program:

$$\inf\{\ \underbrace{c^t x}_{\text{objective}}\ :\ \underbrace{Ax = b}_{\text{linear constraints}}\ ,\ \underbrace{\mathcal{B}_i(x) \succeq 0}_{\text{PSD constraints}}\ \}$$

with x the vector of unknowns, and $\mathcal{B}_i(x)$ the blocks of x.

Solving an SDP: Rage against the machine precision

A semidefinite program:

$$\inf\{ \underbrace{c^t x}_{\text{objective}} \; : \; \underbrace{Ax = b}_{\text{linear constraints}} \; , \; \underbrace{\mathcal{B}_i(x) \succeq 0}_{\text{PSD constraints}} \}$$

with x the vector of unknowns, and $\mathcal{B}_i(x)$ the blocks of x.

- Solving an SDP exactly is sometimes possible (Henrion-Naldi-Safey El Din 2018).

Solving an SDP: Rage against the machine precision

A semidefinite program:

$$\inf\{ \underbrace{c^t x}_{\text{objective}} \ : \ \underbrace{Ax = b}_{\text{linear constraints}} \ , \ \underbrace{\mathcal{B}_i(x) \succeq 0}_{\text{PSD constraints}} \}$$

with x the vector of unknowns, and $\mathcal{B}_i(x)$ the blocks of x.

- Solving an SDP exactly is sometimes possible (Henrion-Naldi-Safey El Din 2018).
- For larger problems, SDP solvers provide approximate solutions in floating point in polynomial time.

Solving an SDP: Rage against the machine precision

A semidefinite program:

$$\inf\{ \underbrace{c^t x}_{\text{objective}} : \underbrace{Ax = b}_{\text{linear constraints}}, \underbrace{\mathcal{B}_i(x) \succeq 0}_{\text{PSD constraints}} \}$$

with x the vector of unknowns, and $\mathcal{B}_i(x)$ the blocks of x.

- Solving an SDP exactly is sometimes possible (Henrion-Naldi-Safey El Din 2018).
- For larger problems, SDP solvers provide approximate solutions in floating point in polynomial time.

How can we turn an approximate solution into an exact one?

Solving an SDP: Rage against the machine precision

A semidefinite program:

$$\inf\{ \underbrace{c^t x}_{\text{objective}} : \underbrace{Ax = b}_{\text{linear constraints}} , \underbrace{\mathcal{B}_i(x) \succeq 0}_{\text{PSD constraints}} \}$$

with x the vector of unknowns, and $\mathcal{B}_i(x)$ the blocks of x.

- Solving an SDP exactly is sometimes possible (Henrion-Naldi-Safey El Din 2018).
- For larger problems, SDP solvers provide approximate solutions in floating point in polynomial time.

How can we turn an approximate solution into an exact one?

- Even if the SDP is defined over $\mathbb{Q}$, optimal solutions can require high algebraic degree (Nie-Ranestad-Sturmfels 2008).

Solving an SDP: Rage against the machine precision

A semidefinite program:

$$\inf\{ \underbrace{c^t x}_{\text{objective}} \;:\; \underbrace{Ax = b}_{\text{linear constraints}} \;,\; \underbrace{\mathcal{B}_i(x) \succeq 0}_{\text{PSD constraints}} \}$$

with x the vector of unknowns, and $\mathcal{B}_i(x)$ the blocks of x.

- Solving an SDP exactly is sometimes possible (Henrion-Naldi-Safey El Din 2018).
- For larger problems, SDP solvers provide approximate solutions in floating point in polynomial time.

How can we turn an approximate solution into an exact one?

- Even if the SDP is defined over $\mathbb{Q}$, optimal solutions can require high algebraic degree (Nie-Ranestad-Sturmfels 2008).
- Our context: The problems provide a candidate field to round over, either $\mathbb{Q}$ or $\mathbb{Q}(\sqrt{d})$.

Rounding over $\mathbb{Q}$: the affine conditions

Solve the SDP numerically in high precision (SDPA-GMP),

$\rightarrow$ get an approximate solution x^*:

Rounding over $\mathbb{Q}$: the affine conditions

Solve the SDP numerically in high precision (SDPA-GMP),

$\rightarrow$ get an approximate solution x^*:

- $Ax^* \approx b$

Rounding over $\mathbb{Q}$: the affine conditions

Solve the SDP numerically in high precision (SDPA-GMP),

$\rightarrow$ get an approximate solution x^*:

- $Ax^* \approx b$
- The blocks $\mathcal{B}_i(x^*)$ might have negative near zero eigenvalues.

Rounding over $\mathbb{Q}$: the affine conditions

Solve the SDP numerically in high precision (SDPA-GMP),

$\to$ get an approximate solution x^*:

- $Ax^* \approx b$
- The blocks $\mathcal{B}_i(x^*)$ might have negative near zero eigenvalues.

We want to find a solution x close to x^* and such that

$$Ax = b.$$

Rounding over $\mathbb{Q}$: the affine conditions

Solve the SDP numerically in high precision (SDPA-GMP),

$\rightarrow$ get an approximate solution x^*:

- $Ax^* \approx b$
- The blocks $\mathcal{B}_i(x^*)$ might have negative near zero eigenvalues.

We want to find a solution x close to x^* and such that

$$Ax = b.$$

- Put the system into reduced row echelon form in rational arithmetic,

Rounding over $\mathbb{Q}$: the affine conditions

Solve the SDP numerically in high precision (SDPA-GMP),

$\rightarrow$ get an approximate solution x^*:

- $Ax^* \approx b$
- The blocks $\mathcal{B}_i(x^*)$ might have negative near zero eigenvalues.

We want to find a solution x close to x^* and such that

$$Ax = b.$$

- Put the system into reduced row echelon form in rational arithmetic,
- Solve the system by backsubstitution. For every free variable, take a value close to the corresponding value in x^*.

Rounding over $\mathbb{Q}$: the affine conditions

Solve the SDP numerically in high precision (SDPA-GMP),

$\rightarrow$ get an approximate solution x^*:

- $Ax^* \approx b$
- The blocks $\mathcal{B}_i(x^*)$ might have negative near zero eigenvalues.

We want to find a solution x close to x^* and such that

$$Ax = b.$$

- Put the system into reduced row echelon form in rational arithmetic,
- Solve the system by backsubstitution. For every free variable, take a value close to the corresponding value in x^*.

The linear system is then satisfied... But what about the PSD conditions?

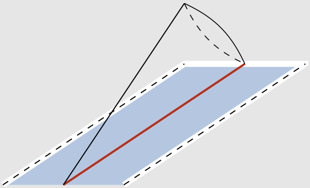
Rounding over $\mathbb{Q}$: the PSD conditions

Rounding over $\mathbb{Q}$: the PSD conditions

- If all the eigenvalues of $\mathcal{B}_i(x^*)$ are far away from zero, $\mathcal{B}_i(x)$ will be positive definite.

Rounding over $\mathbb{Q}$: the PSD conditions

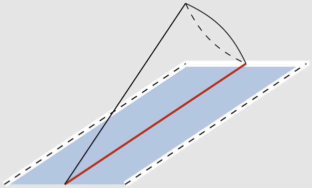
- If all the eigenvalues of $\mathcal{B}_i(x^*)$ are far away from zero, $\mathcal{B}_i(x)$ will be positive definite.



- If the dimension of the affine space is larger than that of the feasible set, we are in trouble. How to deal with near zero eigenvalues?

Rounding over $\mathbb{Q}$: the PSD conditions

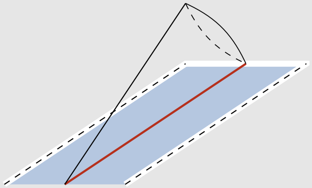
- If all the eigenvalues of $\mathcal{B}_i(x^*)$ are far away from zero, $\mathcal{B}_i(x)$ will be positive definite.



- If the dimension of the affine space is larger than that of the feasible set, we are in trouble. How to deal with near zero eigenvalues?
- Sometimes, zero eigenvalues can be forced by some additional affine constraints coming from an optimal configuration.
  This is sometimes enough... (Cohn-Woo 2012).

Rounding over $\mathbb{Q}$: the PSD conditions

- If all the eigenvalues of $\mathcal{B}_i(x^*)$ are far away from zero, $\mathcal{B}_i(x)$ will be positive definite.



- If the dimension of the affine space is larger than that of the feasible set, we are in trouble. How to deal with near zero eigenvalues?
- Sometimes, zero eigenvalues can be forced by some additional affine constraints coming from an optimal configuration.
  This is sometimes enough... (Cohn-Woo 2012).
- Sometimes not. By undertsanding the kernels, we can force all these constraints!

Rounding over $\mathbb{Q}$: the complete procedure

1. Compute an approximate solution x^*.

Rounding over $\mathbb{Q}$: the complete procedure

1. Compute an approximate solution x^*.
2. Compute the kernels of the $\mathcal{B}_i(x^*)$'s and detect the expected kernels of the $\mathcal{B}_i(x)$'s using LLL.

Rounding over $\mathbb{Q}$: the complete procedure

1. Compute an approximate solution x^*.
2. Compute the kernels of the $\mathcal{B}_i(x^*)$'s and detect the expected kernels of the $\mathcal{B}_i(x)$'s using LLL.
3. Include the new linear constraints in the linear system $Ax = b$.

Rounding over $\mathbb{Q}$: the complete procedure

1. Compute an approximate solution x^*.
2. Compute the kernels of the $\mathcal{B}_i(x^*)$'s and detect the expected kernels of the $\mathcal{B}_i(x)$'s using LLL.
3. Include the new linear constraints in the linear system $Ax = b$.
4. Row reduce the linear system.

Rounding over $\mathbb{Q}$: the complete procedure

1. Compute an approximate solution x^*.
2. Compute the kernels of the $\mathcal{B}_i(x^*)$'s and detect the expected kernels of the $\mathcal{B}_i(x)$'s using LLL.
3. Include the new linear constraints in the linear system $Ax = b$.
4. Row reduce the linear system.
5. Solve it with backsubstitution using x^*.

Rounding over $\mathbb{Q}$: the complete procedure

1. Compute an approximate solution x^*.
2. Compute the kernels of the $\mathcal{B}_i(x^*)$'s and detect the expected kernels of the $\mathcal{B}_i(x)$'s using LLL.
3. Include the new linear constraints in the linear system $Ax = b$.
4. Row reduce the linear system.
5. Solve it with backsubstitution using x^*.
6. Check that the blocks of the rounded solution are indeed PSD.

Rounding over $\mathbb{Q}$: the complete procedure

1. Compute an approximate solution x^*.
2. Compute the kernels of the $\mathcal{B}_i(x^*)$'s and detect the expected kernels of the $\mathcal{B}_i(x)$'s using LLL.
3. Include the new linear constraints in the linear system $Ax = b$.
4. Row reduce the linear system.
5. Solve it with backsubstitution using x^*.
6. Check that the blocks of the rounded solution are indeed PSD.

Thank you!