The Subdivision of Large Simplicial Cones in Normaliz

Richard Sieg

joint with
Winfried Bruns
& Christoph Söger

UNIVERSITÄT OSNABRÜCK

ICMS 2016

SAN FRANCISCO STATE UNIVERSITY · 1899 ·

NORMALIZ

NORMALIZ

- Open source software (GPL)
- written in C++ (using Boost and GMP/MPIR)
- parallelized with OpenMP
- runs under Linux, MacOs and MS Windows
- C++ library libnormaliz
- GUI interface jNormaliz

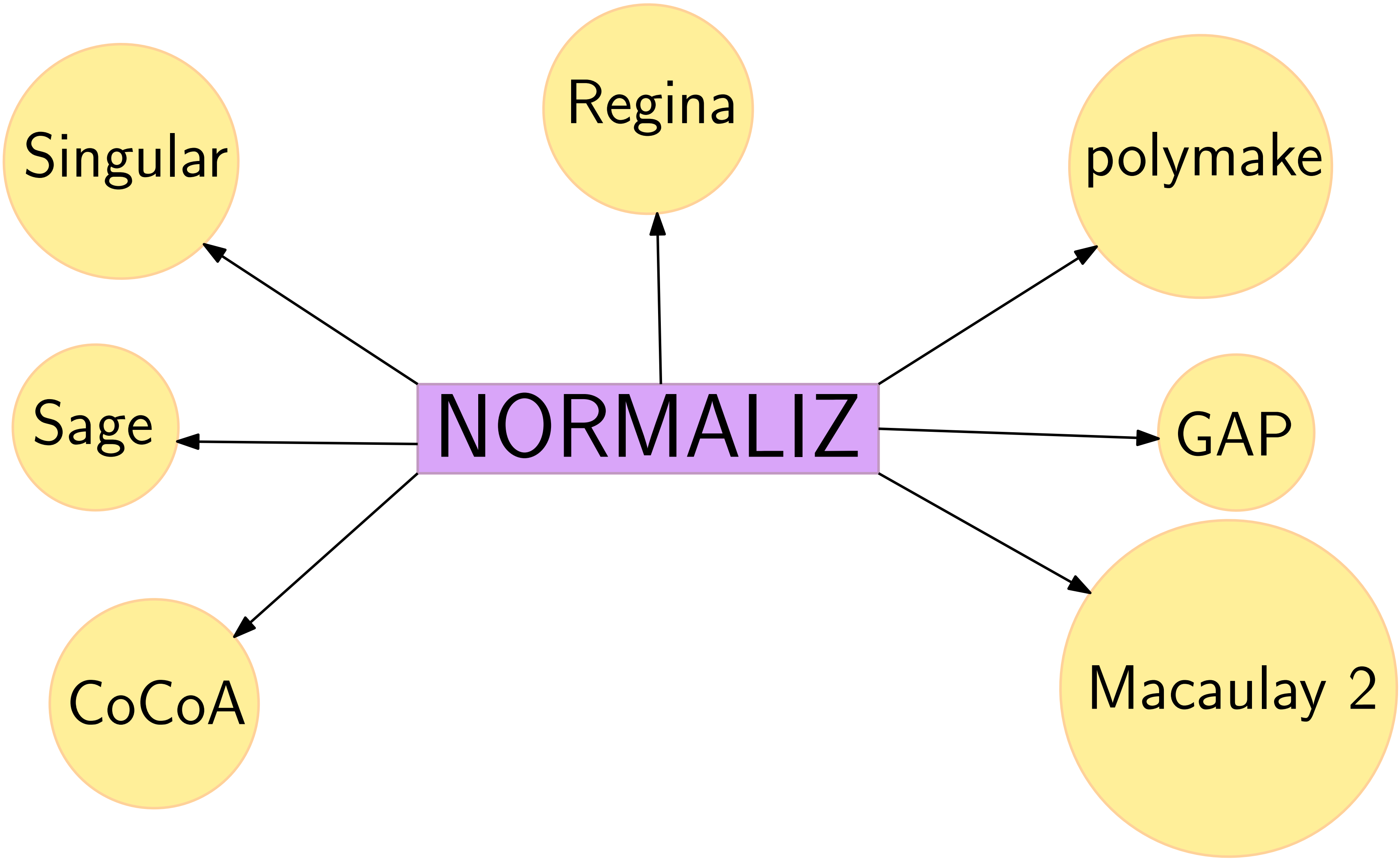
Version 3.1 just released!

http://www.math.uos.de/normaliz

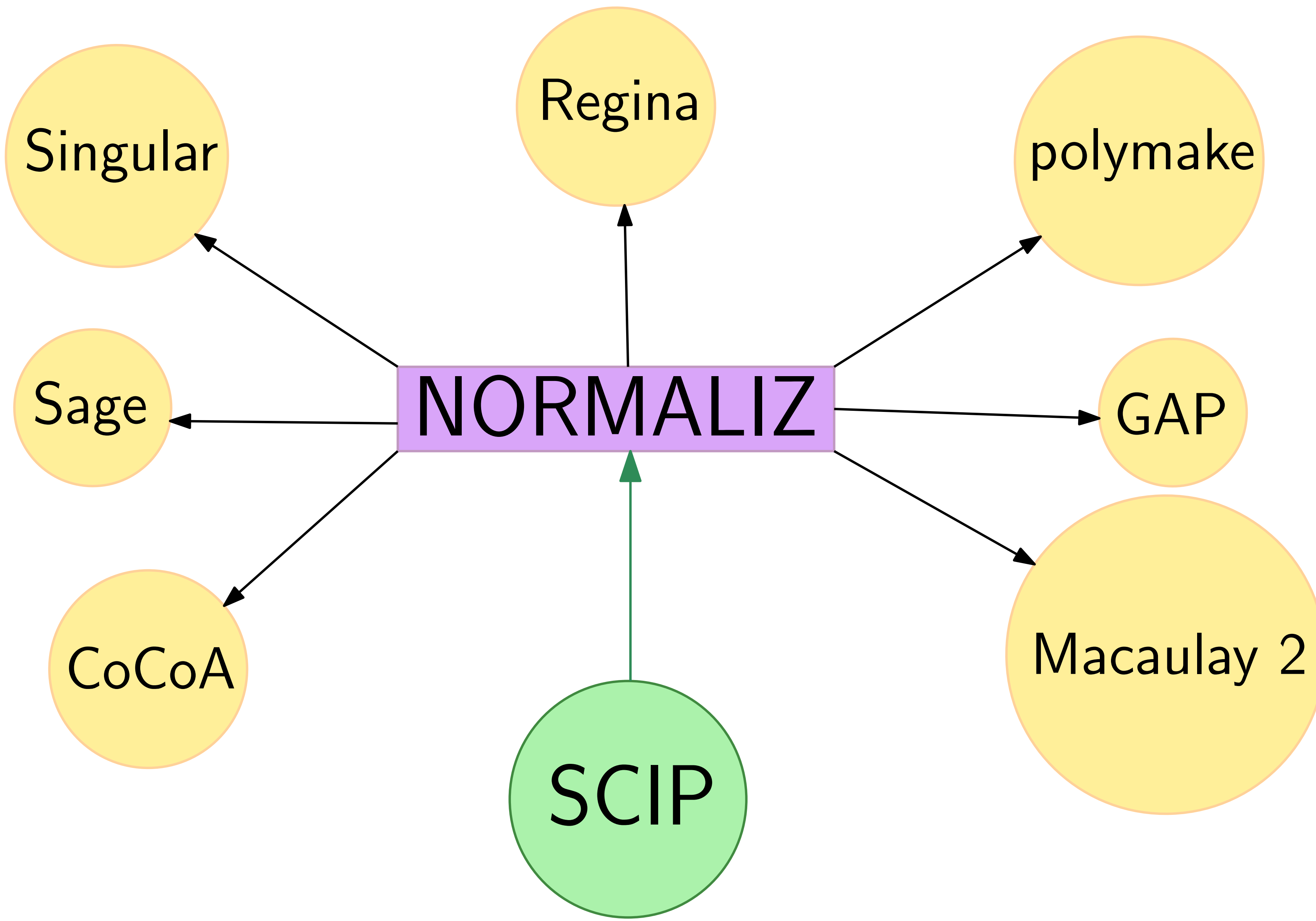
NORMALIZ

- Open source software (GPL)
- written in C++ (using Boost and GMP/MPIR)
- parallelized with OpenMP
- runs under Linux, MacOs and MS Windows
- C++ library libnormaliz
- GUI interface jNormaliz

NORMALIZ

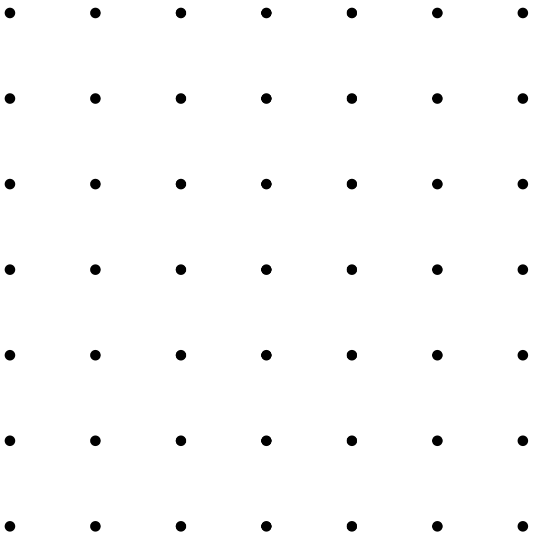


NORMALIZ
Regina
Singular
polymake
Sage
GAP
CoCoA
Macaulay 2


Regina
Singular
polymake
Sage
NORMALIZ
GAP
CoCoA
Macaulay 2
SCIP

Rational Cones

L . . . a lattice (subgroup of $\mathbb{Z}^d$)



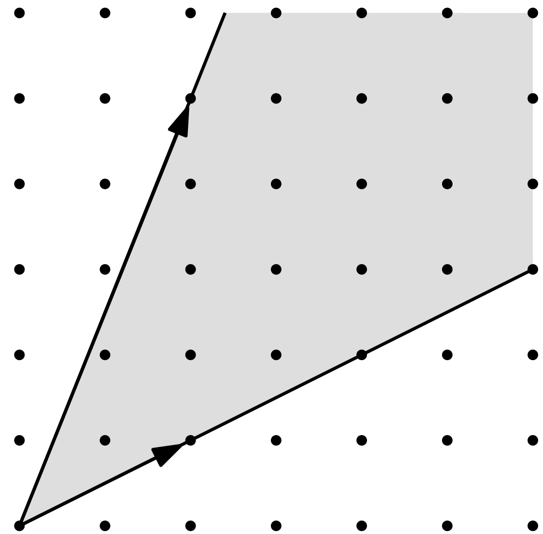
Rational Cones

L . . . a lattice (subgroup of $\mathbb{Z}^d$)

C . . . a (rational polyhedral) cone

$$\begin{aligned} C &= \operatorname{cone}(x_1, \ldots, x_n) \subset \mathbb{R}^d \\ &= \{a_1 x_1 + \cdots + a_n x_n \mid a_1, \ldots, a_n \in \mathbb{R}_+\} \\ &= \{x \in \mathbb{R}^n \mid Ax \geq 0\} \end{aligned}$$

with a generating system $x_1, \ldots, x_n \in \mathbb{Z}^d$.

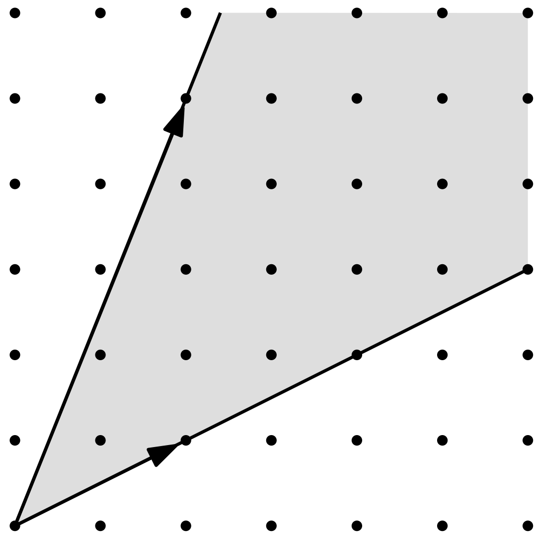


Rational Cones

$L \ldots$ a lattice (subgroup of $\mathbb{Z}^d$)

$C \ldots$ a (rational polyhedral) cone

$$\begin{aligned} C &= \mathrm{cone}(x_1, \ldots, x_n) \subset \mathbb{R}^d \\ &= \{a_1 x_1 + \cdots + a_n x_n \mid a_1, \ldots, a_n \in \mathbb{R}_+\} \\ &= \{x \in \mathbb{R}^n \mid Ax \geq 0\} \end{aligned}$$



with a generating system $x_1, \ldots, x_n \in \mathbb{Z}^d$.

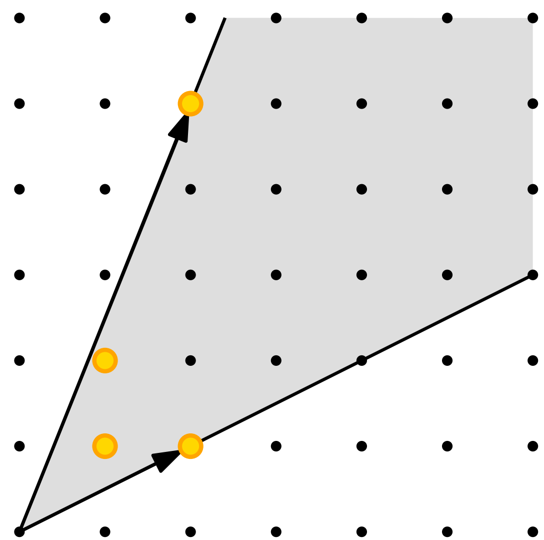
C simplicial: $x_1, \ldots, x_n$ linearly independent

Rational Cones

L . . . a lattice (subgroup of $\mathbb{Z}^d$)

C . . . a (rational polyhedral) cone

$$\begin{aligned} C &= \operatorname{cone}(x_1, \ldots, x_n) \subset \mathbb{R}^d \\ &= \{a_1 x_1 + \cdots + a_n x_n \mid a_1, \ldots, a_n \in \mathbb{R}_+\} \\ &= \{x \in \mathbb{R}^n \mid Ax \geq 0\} \end{aligned}$$



with a generating system $x_1, \ldots, x_n \in \mathbb{Z}^d$.

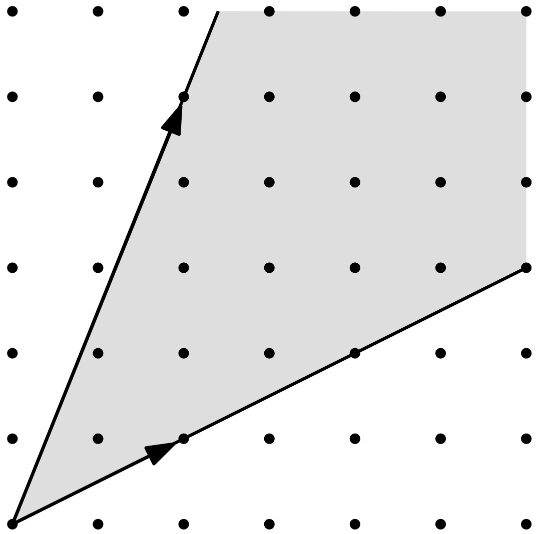
C simplicial: $x_1, \ldots, x_n$ linearly independent

Theorem [Gordan's Lemma]

Let $C \subset \mathbb{R}^d$ be the cone generated by $x_1, \ldots, x_n \in \mathbb{Z}^d$. Then $C \cap L$ is an affine monoid M, i.e. a finitely generated submonoid of $\mathbb{Z}^d$.

The Tasks of Normaliz: Hilbert Basis

Assume C pointed: $x, -x \in C \Rightarrow x = 0$.

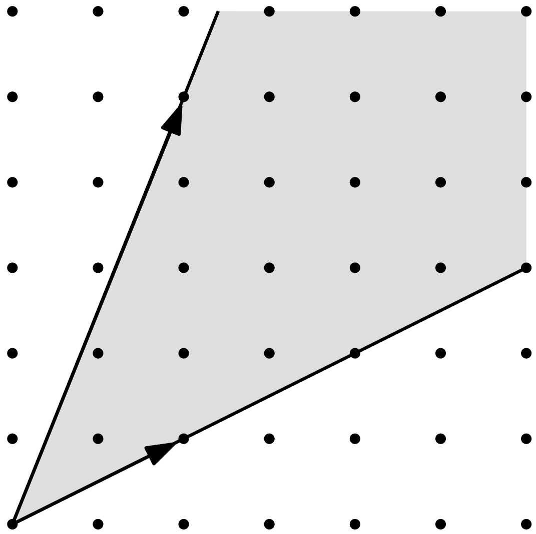


The Tasks of Normaliz: Hilbert Basis

Assume C pointed: $x, -x \in C \;\Rightarrow\; x = 0$.

$x \in M = C \cap L, x \neq 0$ is irreducible:

$$x = y + z \;\Rightarrow\; y = 0 \text{ or } z = 0.$$

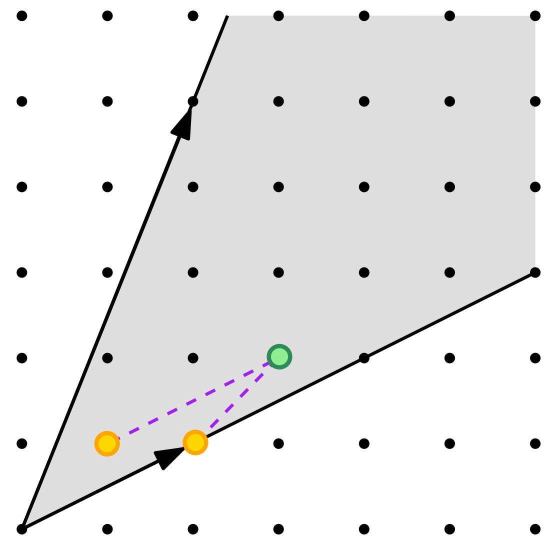


The Tasks of Normaliz: Hilbert Basis

Assume C pointed: $x, -x \in C \;\Rightarrow\; x = 0$.

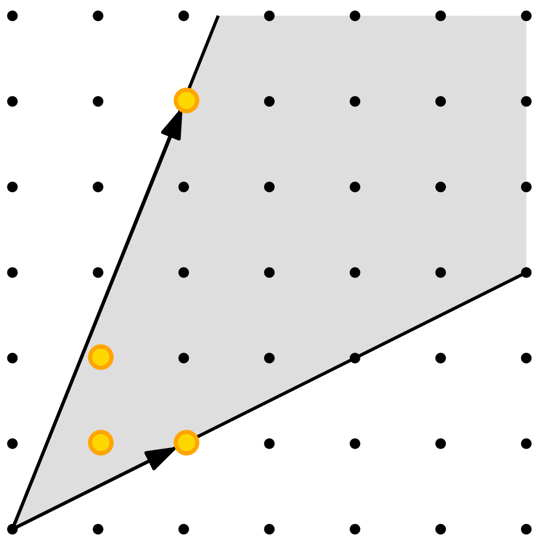
$x \in M = C \cap L, x \neq 0$ is irreducible:

$$x = y + z \;\Rightarrow\; y = 0 \text{ or } z = 0.$$



The Tasks of Normaliz: Hilbert Basis

Assume C pointed: $x, -x \in C \;\Rightarrow\; x = 0$.



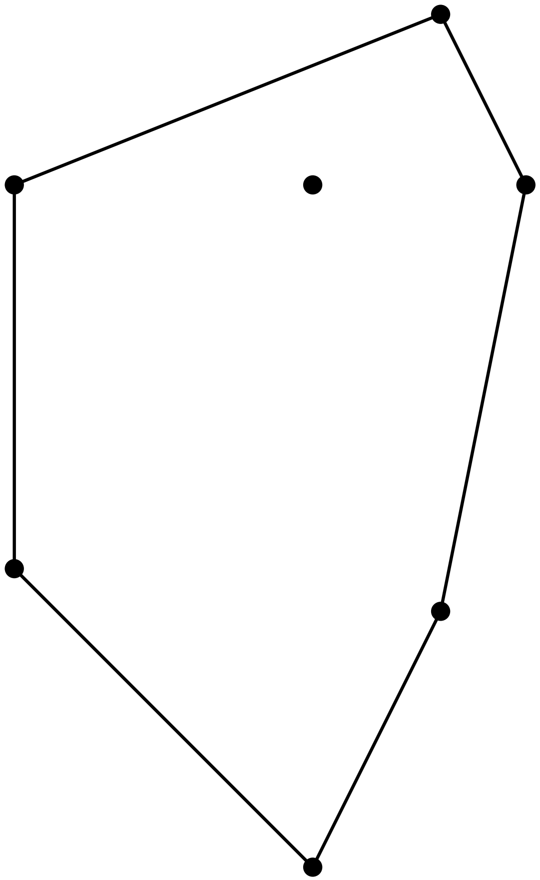
$x \in M = C \cap L, x \neq 0$ is irreducible:

$$x = y + z \;\Rightarrow\; y = 0 \text{ or } z = 0.$$

THEOREM [Hilbert's Basis Theorem]

There are only finitely many irreducible elements in $C \cap L$ and they form the unique minimal system of generators, the Hilbert Basis.

Normaliz Algorithm

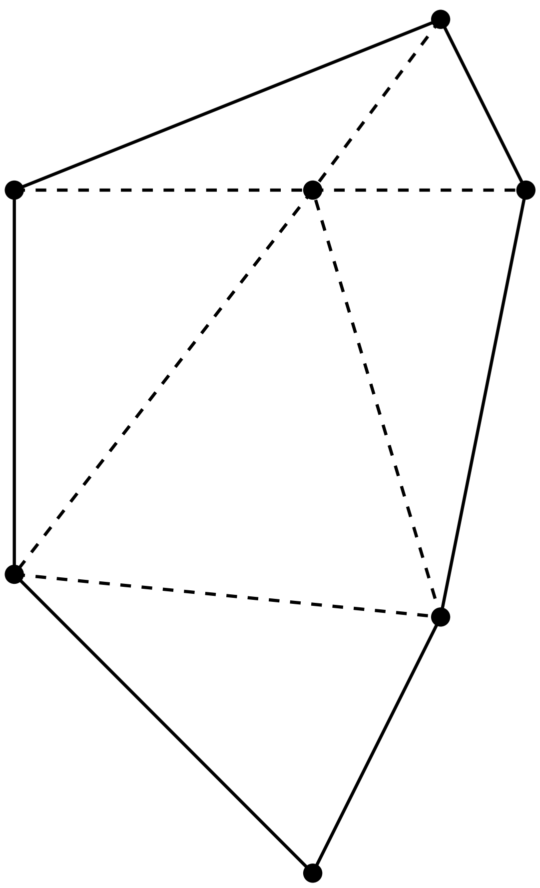


cross section

In the Normaliz algorithm:

- Preparatory coordinate transformation, s.t. the cone is full dimensional and $L = \mathbb{Z}^d$.

Normaliz Algorithm

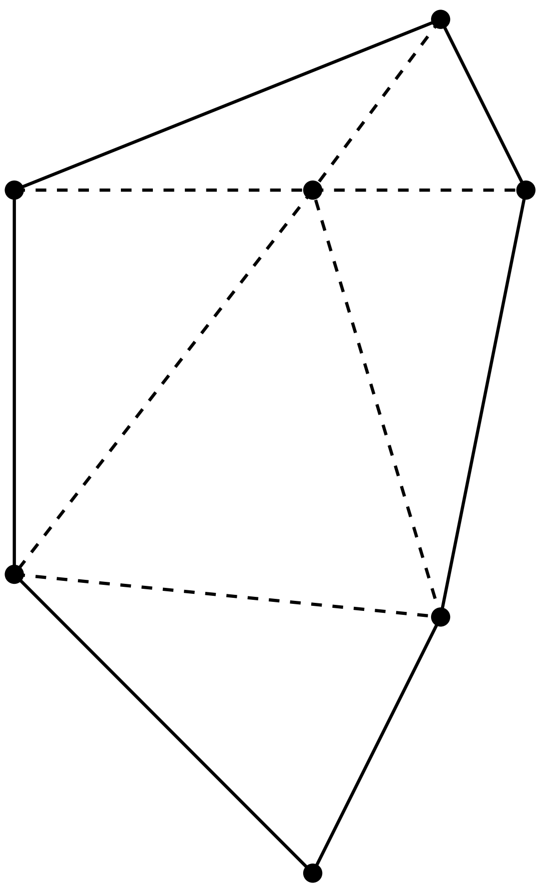


cross section

In the Normaliz algorithm:

- Preparatory coordinate transformation, s.t. the cone is full dimensional and $L = \mathbb{Z}^d$.
- Compute a triangulation of the cone, that is a face-to-face decomposition into simplicial cones. Simplicial cones are generated by linearly independent vectors.

Normaliz Algorithm

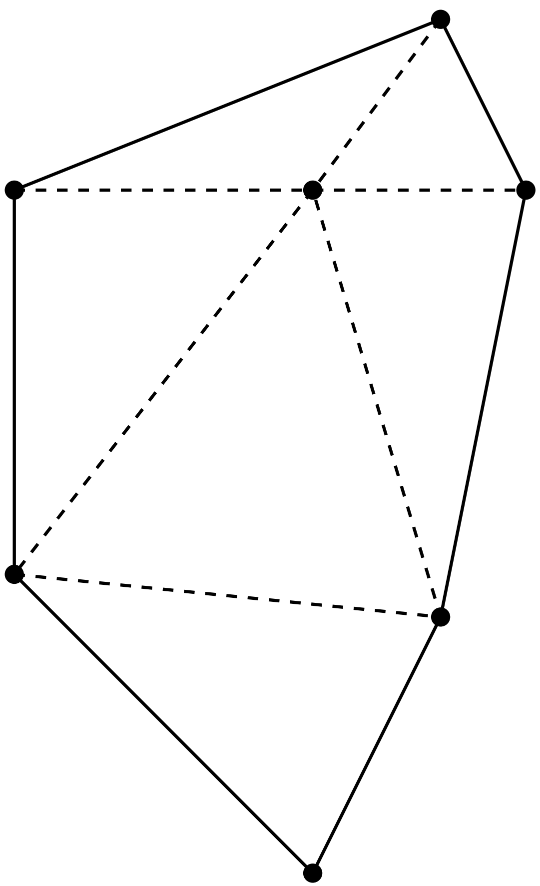


cross section

In the Normaliz algorithm:

- Preparatory coordinate transformation, s.t. the cone is full dimensional and $L = \mathbb{Z}^d$.
- Compute a triangulation of the cone, that is a face-to-face decomposition into simplicial cones. Simplicial cones are generated by linearly independent vectors.
- Evaluate the simplicial cones in the triangulation independently from each other.

Normaliz Algorithm

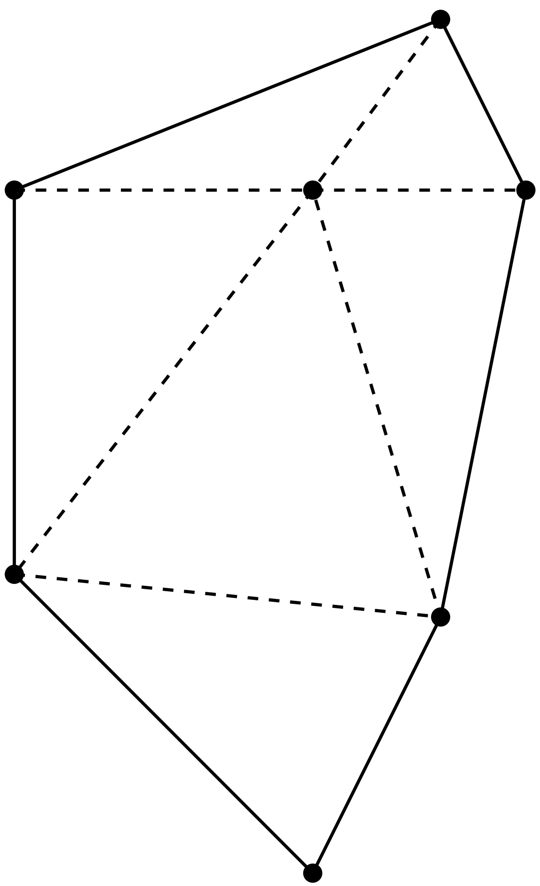


cross section

In the Normaliz algorithm:

- ⋆ Preparatory coordinate transformation, s.t. the cone is full dimensional and $L = \mathbb{Z}^d$.
- ⋆ Compute a triangulation of the cone, that is a face-to-face decomposition into simplicial cones. Simplicial cones are generated by linearly independent vectors.
- ⋆ Evaluate the simplicial cones in the triangulation independently from each other.
- ⋆ Collect the data from the simplicial cones and process it globally.

Normaliz Algorithm

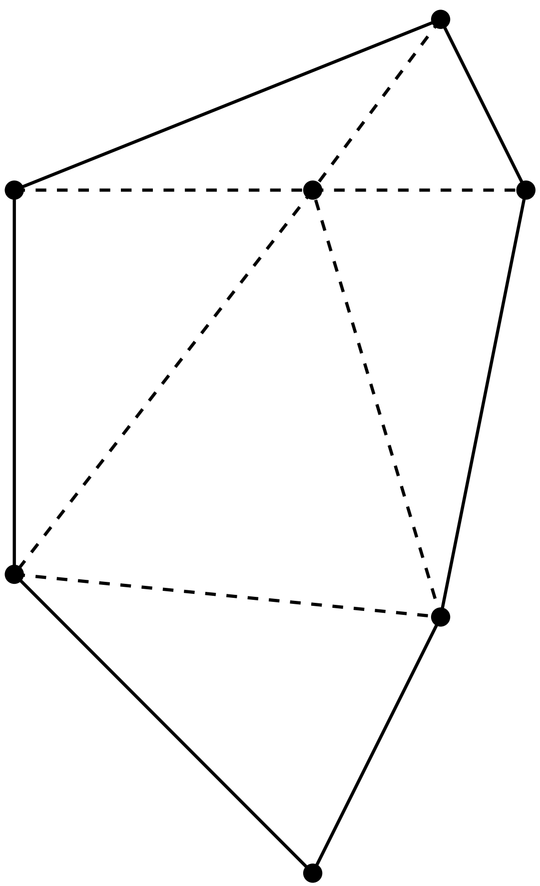


cross section

In the Normaliz algorithm:

- Preparatory coordinate transformation, s.t. the cone is full dimensional and $L = \mathbb{Z}^d$.
- Compute a triangulation of the cone, that is a face-to-face decomposition into simplicial cones. Simplicial cones are generated by linearly independent vectors.
- Evaluate the simplicial cones in the triangulation independently from each other.
- Collect the data from the simplicial cones and process it globally.
- Inverse coordinate transformation.

Normaliz Algorithm



cross section

In the Normaliz algorithm:

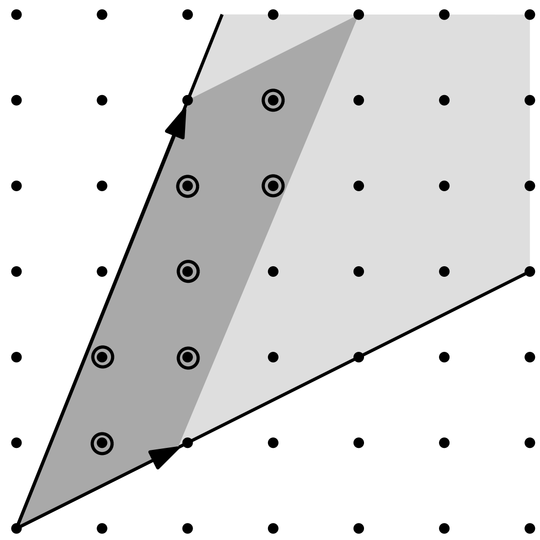
- Preparatory coordinate transformation, s.t. the cone is full dimensional and $L = \mathbb{Z}^d$.
- Compute a triangulation of the cone, that is a face-to-face decomposition into simplicial cones. Simplicial cones are generated by linearly independent vectors.
- Evaluate the simplicial cones in the triangulation independently from each other.
- Collect the data from the simplicial cones and process it globally.
- Inverse coordinate transformation.

Simplicial Cones

$S = \mathrm{cone}(x_1, \ldots, x_d)$ simplex. Then

$$E = \underbrace{\{q_1 x_1 + \cdots + q_d x_d \mid 0 \leq q_i < 1\}}_{\pi} \cap \mathbb{Z}^d$$

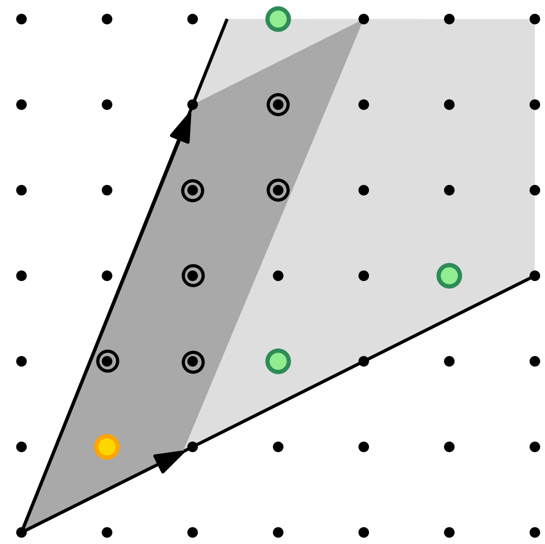
together with $x_1, \ldots, x_d$ generate the monoid $S \cap \mathbb{Z}^d$.



Simplicial Cones

$S = \mathrm{cone}(x_1, \ldots, x_d)$ simplex. Then

$$E = \underbrace{\{q_1 x_1 + \cdots + q_d x_d \mid 0 \le q_i < 1\}}_{\pi} \cap \mathbb{Z}^d$$

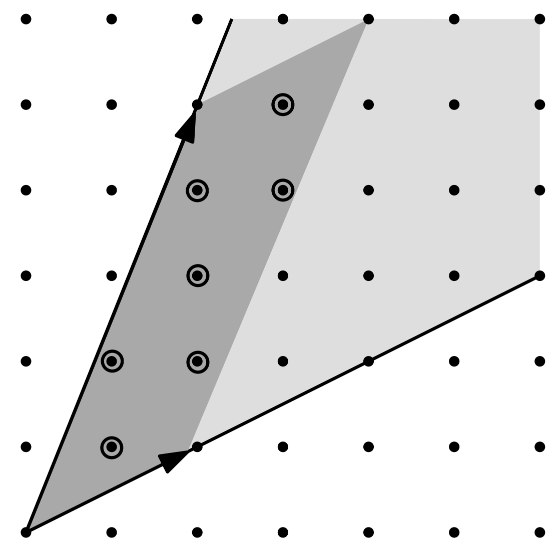


together with $x_1, \ldots, x_d$ generate the monoid $S \cap \mathbb{Z}^d$.

Every residue class in $\mathbb{Z}^d/U$, $U = \mathbb{Z}x_1 + \cdots + \mathbb{Z}x_d$, has exactly one representative in E.

Simplicial Cones

$S = \operatorname{cone}(x_1, \dots, x_d)$ simplex. Then



$$E = \underbrace{\{q_1 x_1 + \cdots + q_d x_d \mid 0 \le q_i < 1\}}_{\pi} \cap \mathbb{Z}^d$$

together with $x_1, \dots, x_d$ generate the monoid $S \cap \mathbb{Z}^d$.

Every residue class in $\mathbb{Z}^d/U$, $U = \mathbb{Z}x_1 + \cdots + \mathbb{Z}x_d$, has exactly one representative in E.

Normaliz generates the points in E. They are candidates for the Hilbert Basis and their number is given by the volume of the simplex

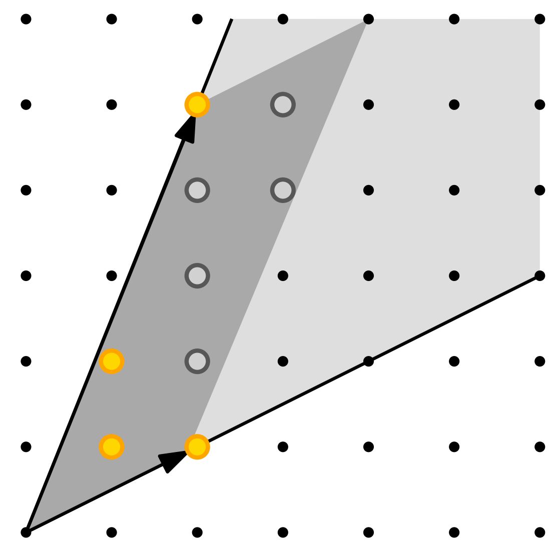
$$|E| = \operatorname{vol}(S) = \det(x_1, \dots, x_d).$$

The points in E are then reduced to a Hilbert Basis of $S \cap \mathbb{Z}^d$.

Simplicial Cones

$S = \text{cone}(x_1, \ldots, x_d)$ simplex. Then

$$E = \underbrace{\{q_1 x_1 + \cdots + q_d x_d \mid 0 \le q_i < 1\}}_{\pi} \cap \mathbb{Z}^d$$



together with $x_1, \ldots, x_d$ generate the monoid $S \cap \mathbb{Z}^d$.

Every residue class in $\mathbb{Z}^d/U$, $U = \mathbb{Z}x_1 + \cdots + \mathbb{Z}x_d$, has exactly one representative in E.

Normaliz generates the points in E. They are candidates for the Hilbert Basis and their number is given by the volume of the simplex

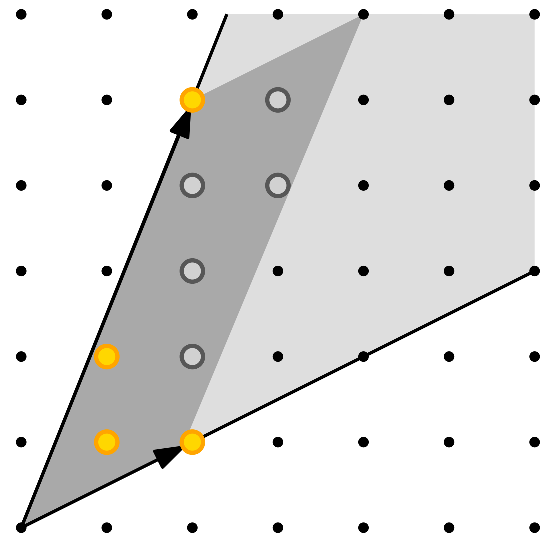
$$|E| = \text{vol}(S) = \det(x_1, \ldots, x_d).$$

The points in E are then reduced to a Hilbert Basis of $S \cap \mathbb{Z}^d$.

Simplicial Cones

$S = \mathrm{cone}(x_1, \ldots, x_d)$ simplex. Then

$$E = \underbrace{\{q_1 x_1 + \cdots + q_d x_d \mid 0 \le q_i < 1\}}_{\pi} \cap \mathbb{Z}^d$$



together with $x_1, \ldots, x_d$ generate the monoid $S \cap \mathbb{Z}^d$.

Every residue class in $\mathbb{Z}^d/U$, $U = \mathbb{Z}x_1 + \cdots + \mathbb{Z}x_d$, has exactly one representative in E.

Normaliz generates the points in E. They are candidates for the Hilbert Basis and their number is given by the volume of the simplex

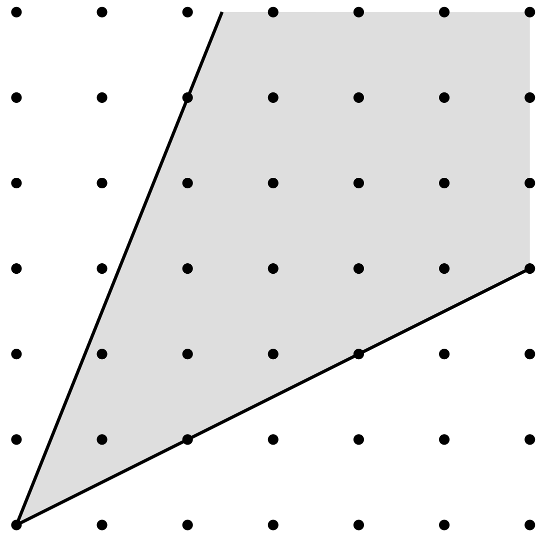
$$|E| = \mathrm{vol}(S) = \det(x_1, \ldots, x_d).$$

The points in E are then reduced to a Hilbert Basis of $S \cap \mathbb{Z}^d$.

Therefore $\mathrm{vol}(S)$ is a critical size for the runtime of Normaliz.

Our Approach

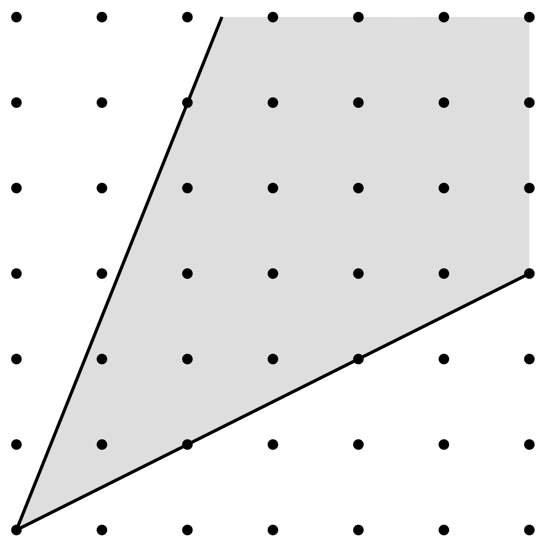
If simplex S has big volume: decompose it into smaller simplices, such that the sum of their volumes decreases remarkably.



Our Approach

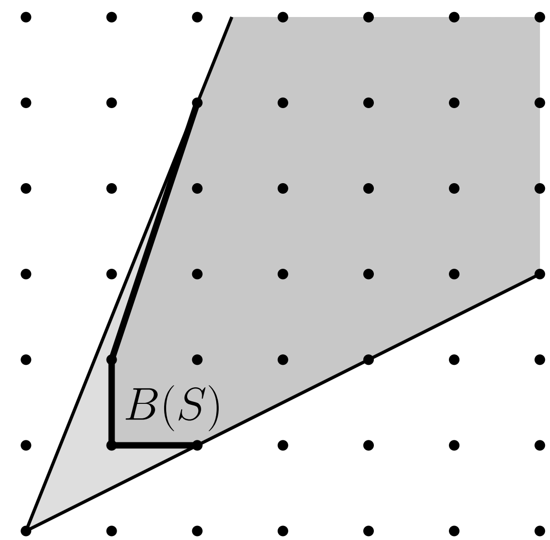
If simplex S has big volume: decompose it into smaller simplices, such that the sum of their volumes decreases remarkably.

How? Compute points from the cone and use them for a new triangulation.



Our Approach

If simplex S has big volume: decompose it into smaller simplices, such that the sum of their volumes decreases remarkably.

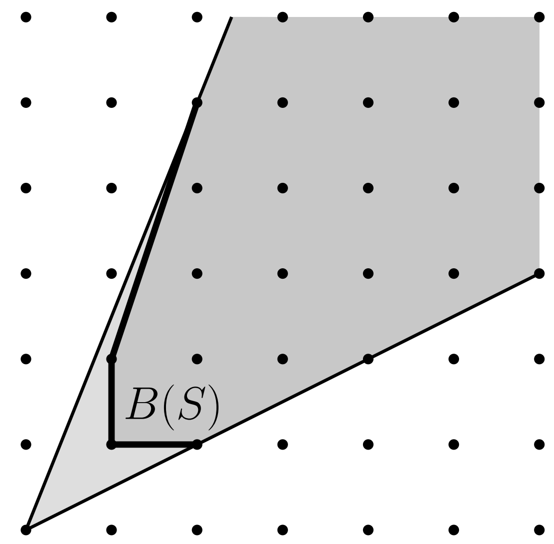


How? Compute points from the cone and use them for a new triangulation.

(Theoretically) Best choice for these points are the vertices of the bottom $B(S)$ (union of the bounded faces of $\mathrm{conv}((S \cap \mathbb{Z}^d) \setminus \{0\})$)

Our Approach

If simplex S has big volume: decompose it into smaller simplices, such that the sum of their volumes decreases remarkably.



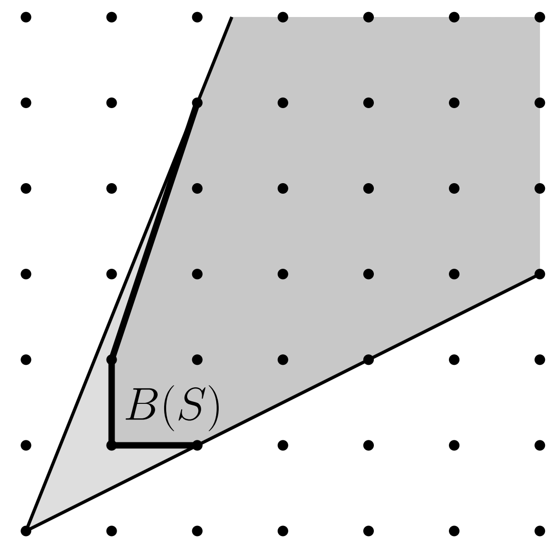
How? Compute points from the cone and use them for a new triangulation.

(Theoretically) Best choice for these points are the vertices of the bottom $B(S)$ (union of the bounded faces of $\mathrm{conv}((S \cap \mathbb{Z}^d) \setminus \{0\})$)

(Practically) Computation of the whole bottom would equalize the benefit from the small volume or even make it worse

Our Approach

If simplex S has big volume: decompose it into smaller simplices, such that the sum of their volumes decreases remarkably.



How? Compute points from the cone and use them for a new triangulation.

(Theoretically) Best choice for these points are the vertices of the bottom $B(S)$ (union of the bounded faces of $\mathrm{conv}((S \cap \mathbb{Z}^d) \setminus \{0\})$)

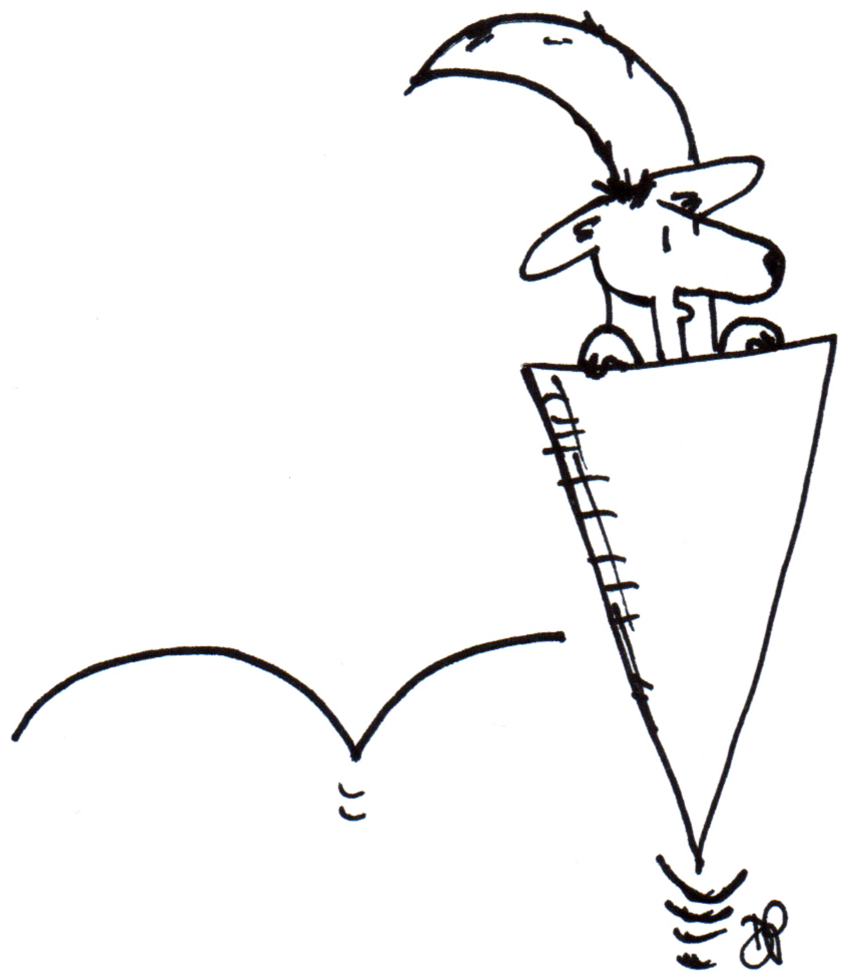
(Practically) Computation of the whole bottom would equalize the benefit from the small volume or even make it worse

Determine only some points from $B(S)$ using

1. Integer Programming

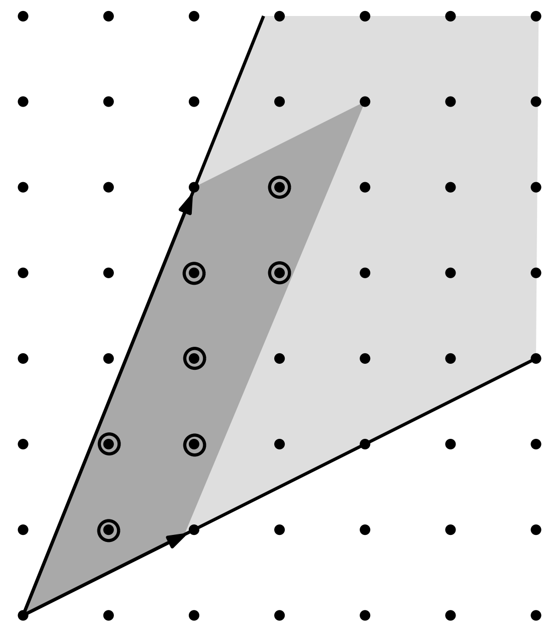
2. Approximation

Integer Programming



The Algorithm

$S = \mathrm{cone}(x_1, \ldots, x_d)$ simplex in triangulation



The Algorithm

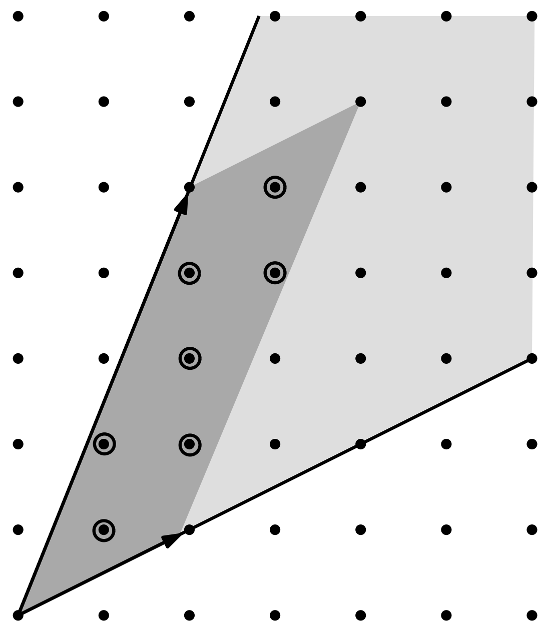
$S = \mathrm{cone}(x_1, \ldots, x_d)$ simplex in triangulation

GOAL

Compute a point x that minimizes the sum of determinants:

$$\sum_{i=1}^{d} \det(x_1, \ldots, x_{i-1}, x, x_{i+1}, \ldots, x_d) = N^T x,$$

N … normal vector on the hyperplane spanned by $x_1, \ldots, x_d$.



The Algorithm

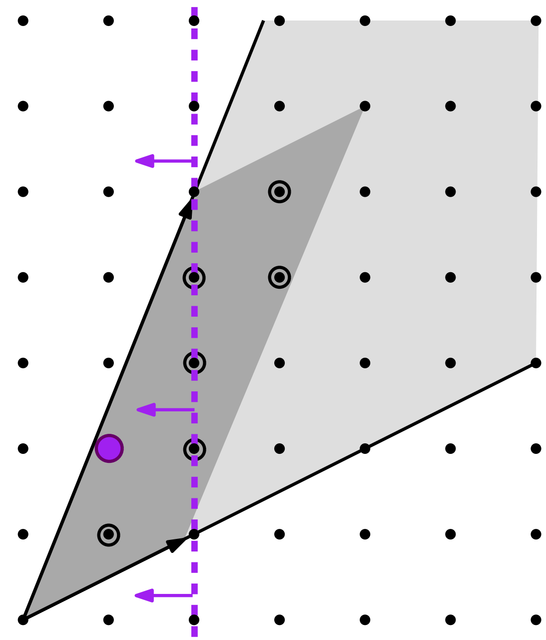
$S = \mathrm{cone}(x_1, \ldots, x_d)$ simplex in triangulation

GOAL

Compute a point x that minimizes the sum of determinants:

$$\sum_{i=1}^{d} \det(x_1, \ldots, x_{i-1}, x, x_{i+1}, \ldots, x_d) = N^T x,$$

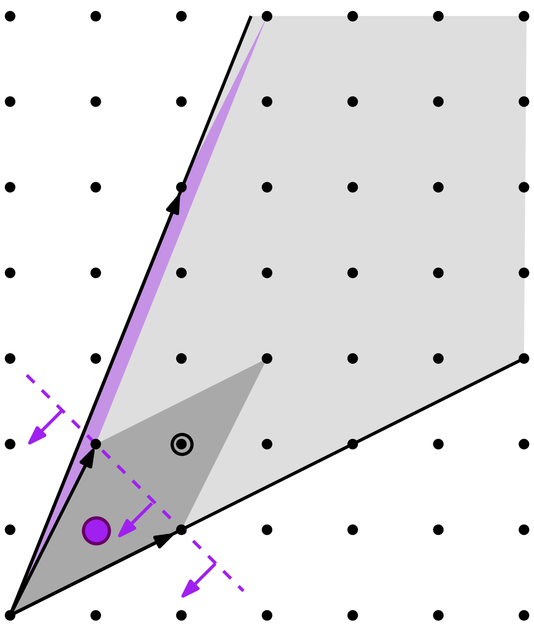
N ... normal vector on the hyperplane spanned by $x_1, \ldots, x_d$.



Solve the IP

$$\boxed{\min\{N^T x \mid x \in S \cap \mathbb{Z}^d, x \neq 0, N^T x < N^T x_1\}} \qquad (\star)$$

The Algorithm



$S = \operatorname{cone}(x_1, \dots, x_d)$ simplex in triangulation

GOAL

Compute a point x that minimizes the sum of determinants:

$$\sum_{i=1}^{d} \det(x_1, \dots, x_{i-1}, x, x_{i+1}, \dots, x_d) = N^T x,$$

N ... normal vector on the hyperplane spanned by $x_1, \dots, x_d$.

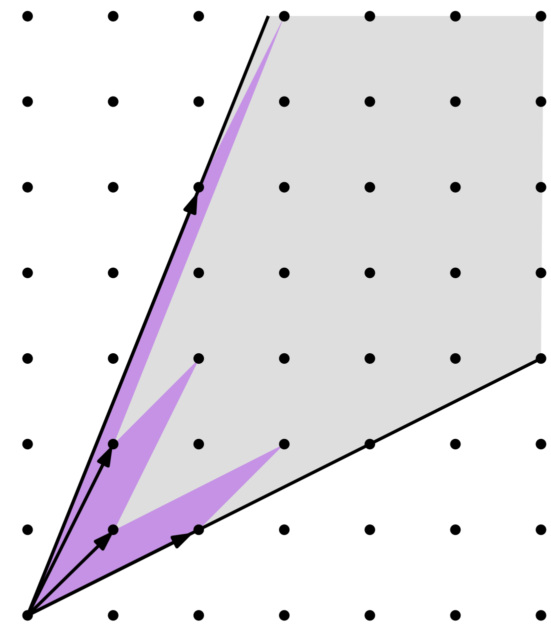
Solve the IP

$$\boxed{\min\{N^T x \mid x \in S \cap \mathbb{Z}^d, x \neq 0, N^T x < N^T x_1\}} \qquad (\star)$$

If problem can be solved: form a stellar subdivision with the solution.

The Algorithm

$S = \mathrm{cone}(x_1, \ldots, x_d)$ simplex in triangulation



GOAL

Compute a point x that minimizes the sum of determinants:

$$\sum_{i=1}^{d} \det(x_1, \ldots, x_{i-1}, x, x_{i+1}, \ldots, x_d) = N^T x,$$

N ... normal vector on the hyperplane spanned by $x_1, \ldots, x_d$.

Solve the IP

$$\boxed{\min\{N^T x \mid x \in S \cap \mathbb{Z}^d, x \neq 0, N^T x < N^T x_1\}} \qquad (\star)$$

If problem can be solved: form a stellar subdivision with the solution.

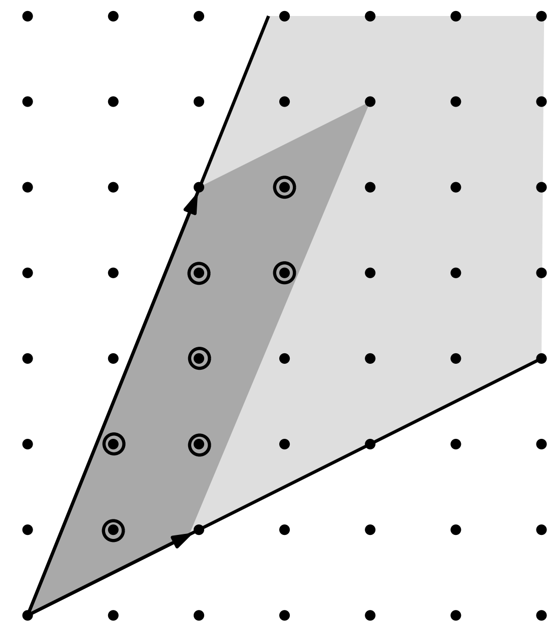
The Algorithm

Algorithm Bottom Points

Input: $S = \operatorname{cone}(x_1, \ldots, x_d)$ simplex with $\operatorname{vol}(S) \geq \text{Bound}$

Return: Points from $B(S)$

1: $\mathcal{B}, \mathcal{S} \leftarrow \emptyset$
2: store S into $\mathcal{S}$
3: **while** $\mathcal{S} \neq \emptyset$ **do**
4: let $T = \operatorname{cone}(y_1, \ldots, y_d)$ be the first element of $\mathcal{S}$ and delete it
5: compute a normal vector N on hyperplane spanned by $y_1, \ldots, y_d$
6: compute hyperplanes $\{H_1, \ldots, H_d\}$ and volume of T
7: **if** $\operatorname{vol}(T) < \text{Bound}$ **then continue**



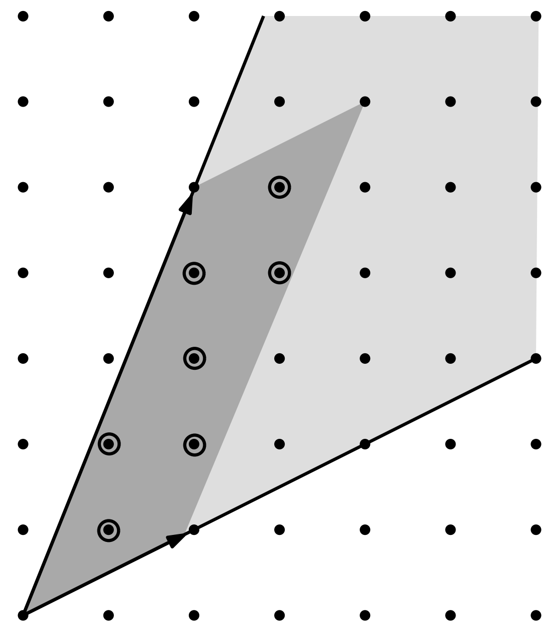
The Algorithm

Algorithm Bottom Points

Input: $S = \mathrm{cone}(x_1, \ldots, x_d)$ simplex with $\mathrm{vol}(S) \geq \mathrm{Bound}$

Return: Points from $B(S)$

1: $\mathcal{B}, \mathcal{S} \leftarrow \emptyset$
2: store S into $\mathcal{S}$
3: **while** $\mathcal{S} \neq \emptyset$ **do**
4: let $T = \mathrm{cone}(y_1, \ldots, y_d)$ be the first element of $\mathcal{S}$ and delete it
5: compute a normal vector N on hyperplane spanned by $y_1, \ldots, y_d$
6: compute hyperplanes $\{H_1, \ldots, H_d\}$ and volume of T
7: **if** $\mathrm{vol}(T) < \mathrm{Bound}$ **then continue**



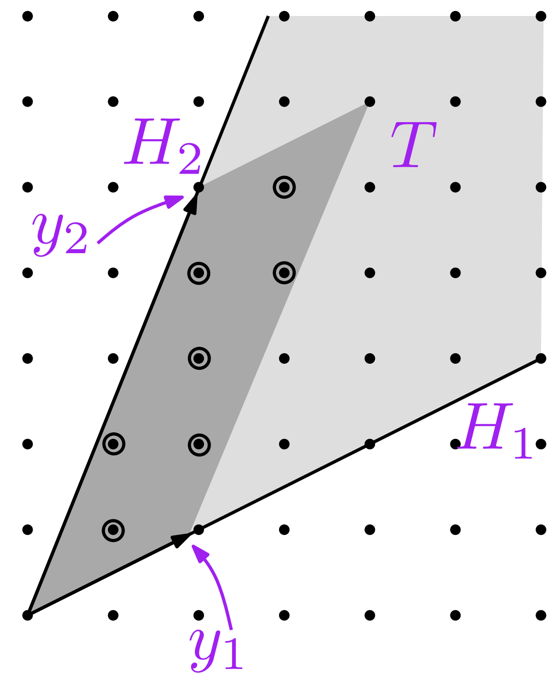
The Algorithm

Algorithm Bottom Points

Input: $S = \mathrm{cone}(x_1, \ldots, x_d)$ simplex with $\mathrm{vol}(S) \geq \mathrm{Bound}$

Return: Points from $B(S)$

$\mathcal{B}, \mathcal{S} \leftarrow \emptyset$
store S into $\mathcal{S}$
while $\mathcal{S} \neq \emptyset$ **do**
    let $\boxed{T = \mathrm{cone}(y_1, \ldots, y_d)}$ be the first element of $\mathcal{S}$ and delete it
    compute a normal vector N on hyperplane spanned by $y_1, \ldots, y_d$
    compute hyperplanes $\boxed{\{H_1, \ldots, H_d\}}$ and volume of T
    if $\mathrm{vol}(T) < \mathrm{Bound}$ **then continue**



The Algorithm

Algorithm Bottom Points

while $\mathcal{S} \neq \emptyset$ **do**

$\vdots$

    if IP $(\star)$ is solvable for T **then**

        $y \leftarrow$ optimal solution of $(\star)$

        store y into $\mathcal{B}$

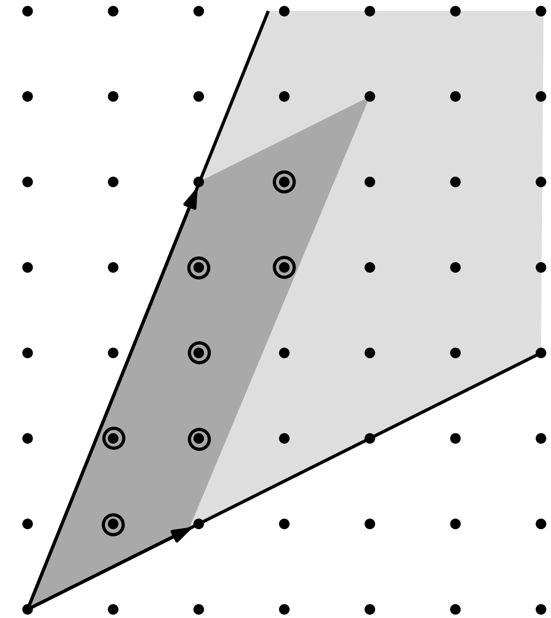
        for all hyperplanes H_i of T **do**

            if $y \notin H_i$ **then**

      $T_i \leftarrow \mathrm{cone}(y_1, \ldots, y_{i-1}, y, y_{i+1}, \ldots, y_d)$

                store T_i into $\mathcal{S}$

return $\mathcal{B}$



$$\boxed{\min\{N^T x \mid x \in S \cap \mathbb{Z}^d, x \neq 0, N^T x < N^T x_1\}} \qquad (\star)$$

The Algorithm

Algorithm Bottom Points

while $\mathcal{S} \neq \emptyset$ **do**

$\vdots$

  if IP $(\star)$ is solvable for T **then**

    $y \leftarrow$ optimal solution of $(\star)$

    store y into $\mathcal{B}$

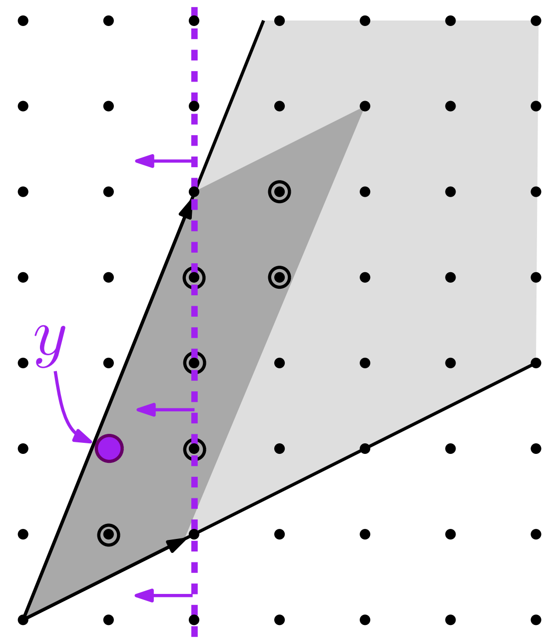
    for all hyperplanes H_i of T **do**

      if $y \notin H_i$ **then**

        $T_i \leftarrow \mathrm{cone}(y_1, \ldots, y_{i-1}, y, y_{i+1}, \ldots, y_d)$

        store T_i into $\mathcal{S}$

return $\mathcal{B}$



$\mathcal{B} = \{(1,2)$

$$\min\{N^T x \mid x \in S \cap \mathbb{Z}^d, x \neq 0, N^T x < N^T x_1\} \qquad (\star)$$

The Algorithm

Algorithm Bottom Points

while $\mathcal{S} \neq \emptyset$ **do**

$\vdots$

  if IP $(\star)$ is solvable for T **then**

    $y \leftarrow$ optimal solution of $(\star)$

    store y into $\mathcal{B}$

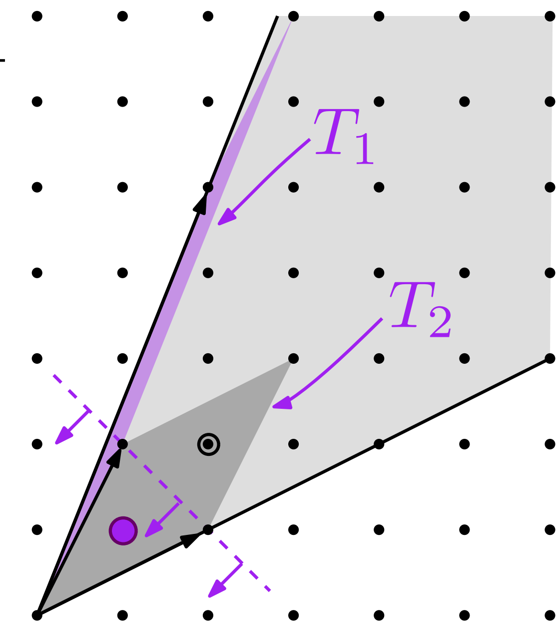
    for all hyperplanes H_i of T **do**

      if $y \notin H_i$ **then**

        $T_i \leftarrow \mathrm{cone}(y_1, \ldots, y_{i-1}, y, y_{i+1}, \ldots, y_d)$

        store T_i into $\mathcal{S}$

return $\mathcal{B}$



$$\mathcal{B} = \{(1,2), (1,1)\}$$

$$\min\{N^T x \mid x \in S \cap \mathbb{Z}^d, x \neq 0, N^T x < N^T x_1\} \qquad (\star)$$

The Algorithm

Algorithm Bottom Points

while $\mathcal{S} \neq \emptyset$ **do**

$\vdots$

  if IP $(\star)$ is solvable for T **then**

    $y \leftarrow$ optimal solution of $(\star)$

    store y into $\mathcal{B}$

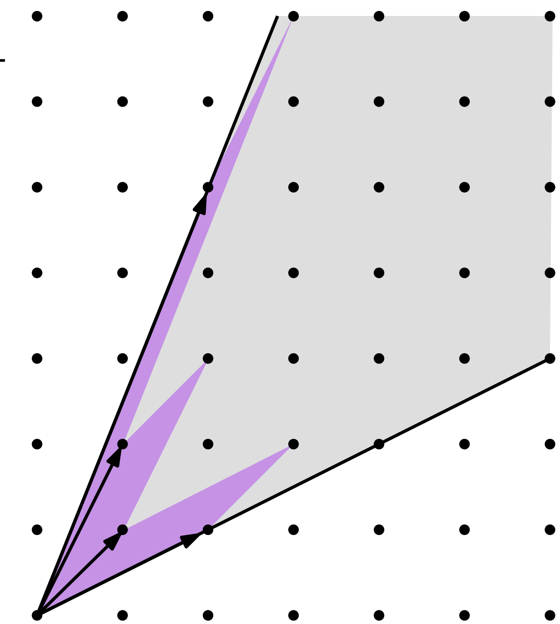
    for all hyperplanes H_i of T **do**

      if $y \notin H_i$ **then**

        $T_i \leftarrow \mathrm{cone}(y_1, \ldots, y_{i-1}, y, y_{i+1}, \ldots, y_d)$

        store T_i into $\mathcal{S}$

return $\mathcal{B}$



$$\mathcal{B} = \{(1,2),(1,1)\}$$

$$\boxed{\min\{N^T x \mid x \in S \cap \mathbb{Z}^d, x \neq 0, N^T x < N^T x_1\}} \qquad (\star)$$

The Algorithm

Algorithm Bottom Points

while $\mathcal{S} \neq \emptyset$ **do**

$\vdots$

  if IP $(\star)$ is solvable for T **then**

    $y \leftarrow$ optimal solution of $(\star)$

    store y into $\mathcal{B}$

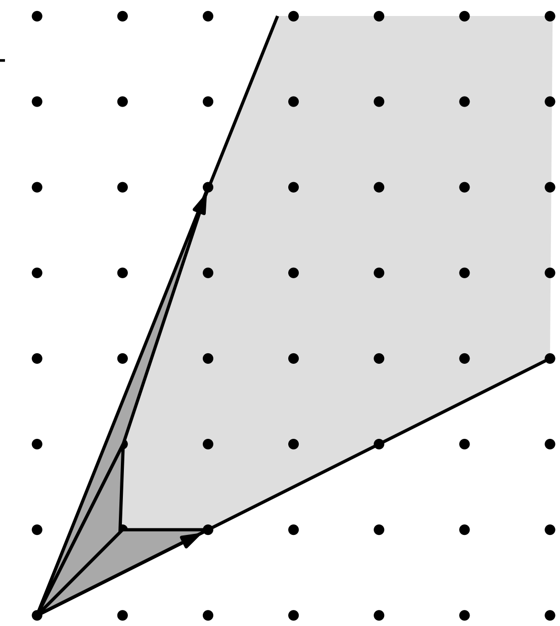
    for all hyperplanes H_i of T **do**

      if $y \notin H_i$ **then**

        $T_i \leftarrow \mathrm{cone}(y_1, \dots, y_{i-1}, y, y_{i+1}, \dots, y_d)$

        store T_i into $\mathcal{S}$

return $\mathcal{B}$



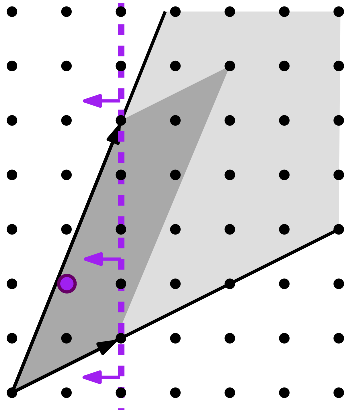
$\mathcal{B} = \{(1,2),(1,1)\}$

$$\min\{N^T x \mid x \in S \cap \mathbb{Z}^d, x \neq 0, N^T x < N^T x_1\} \qquad (\star)$$

We triangulate the lower facets of $\mathrm{conv}(\mathcal{B} \cup \{x_1, \dots, x_d\})$ and evaluate this triangulation with the usual Normaliz algorithm.

The Algorithm

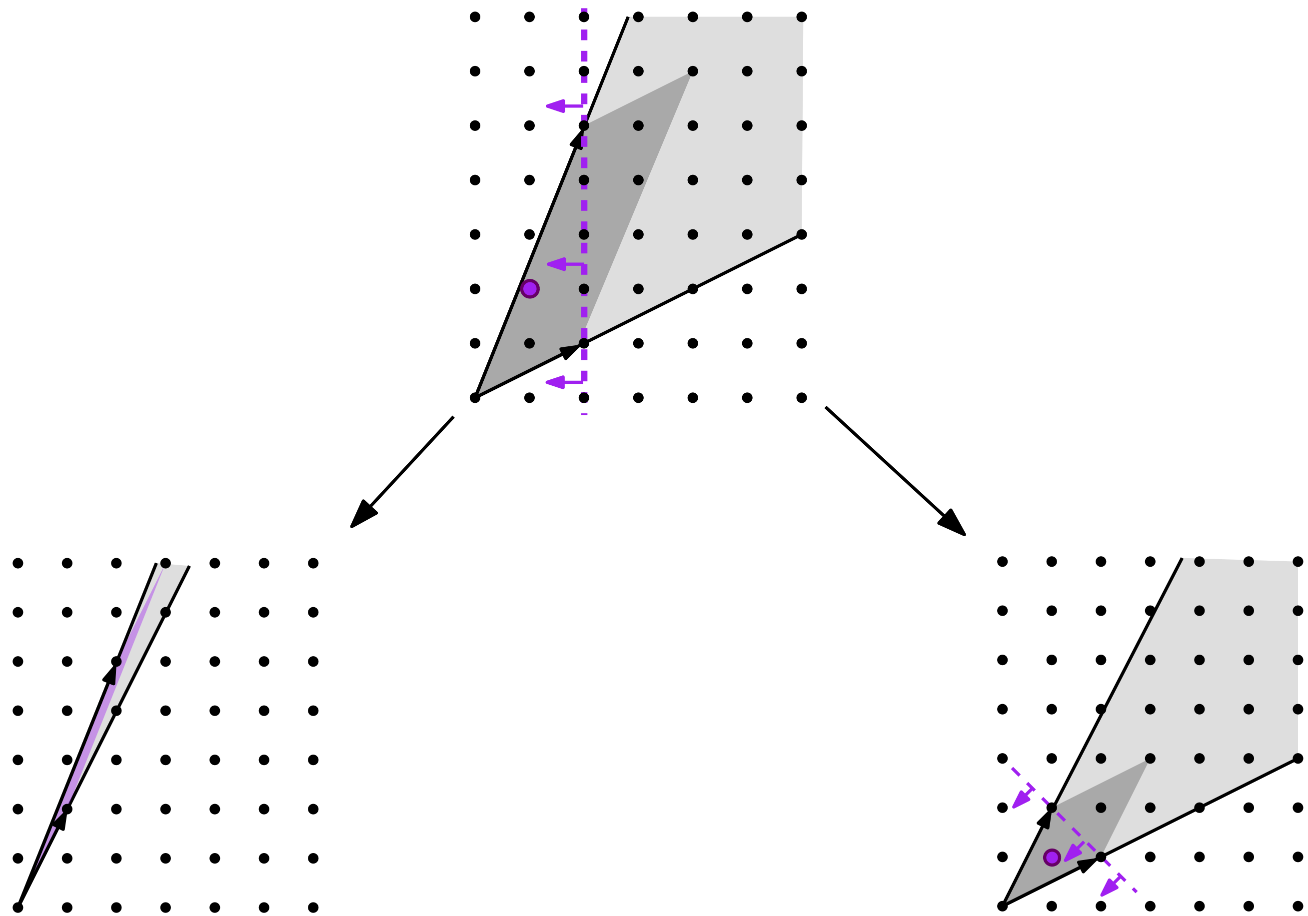
Level 0



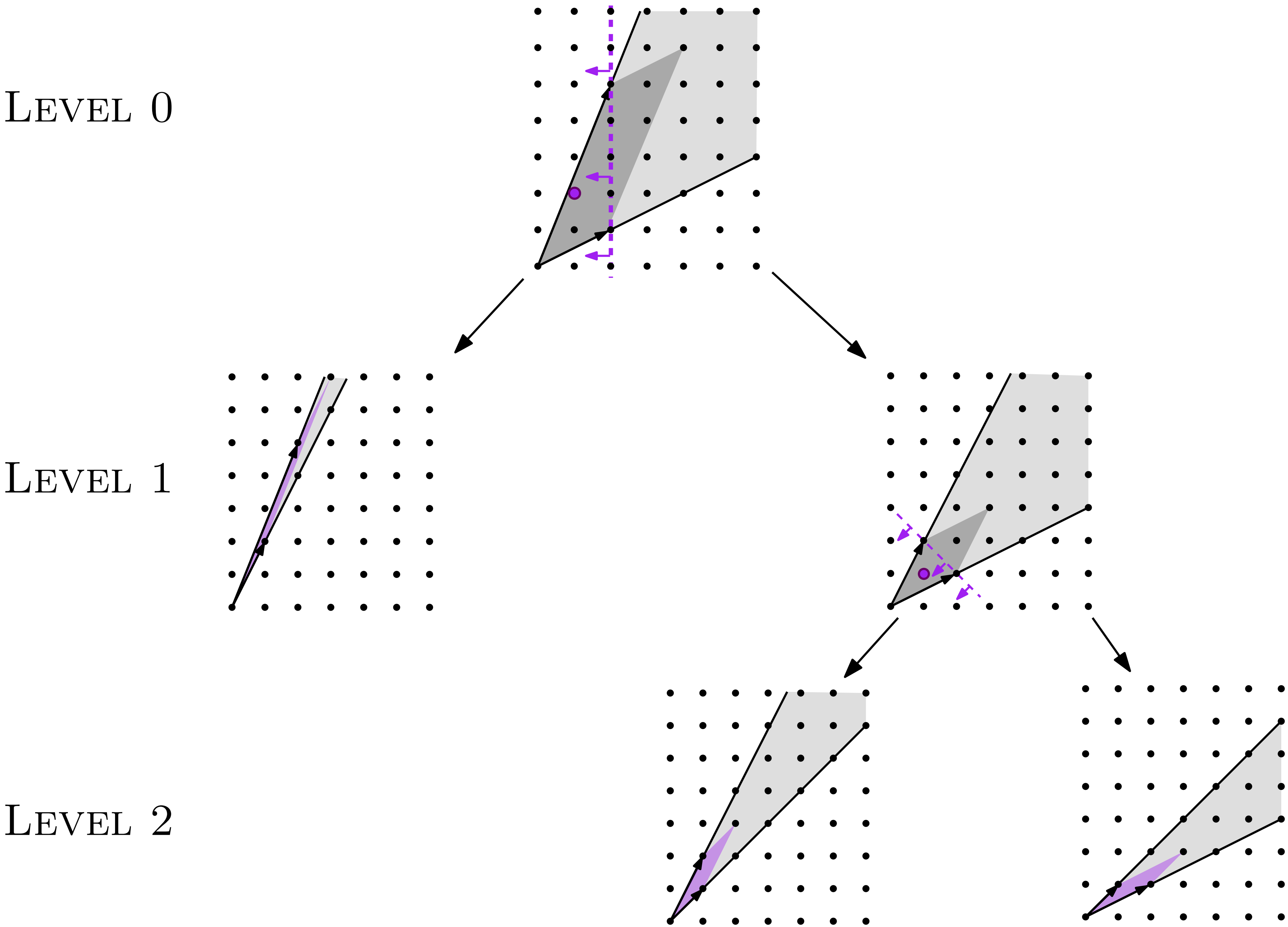
The Algorithm

Level 0

Level 1

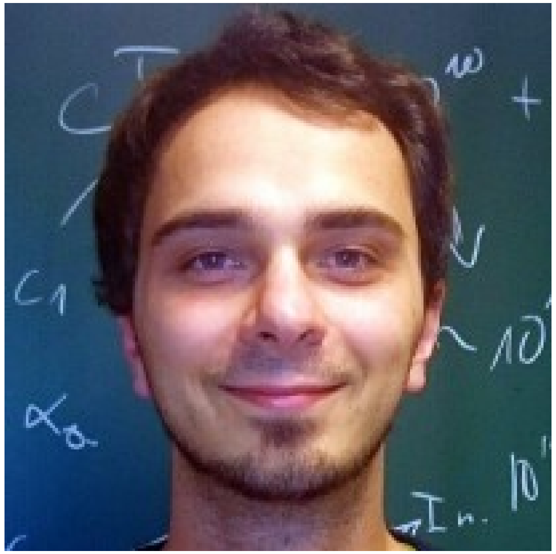


The Algorithm



Implementation & Results

- use SCIP (3.2.0) via its C++ interace



Gregor Hendel

Implementation & Results

- use SCIP (3.2.0) via its C++ interace
- parallelization with OpenMP
  - individual time limit
  - individual feasibility bounds

Implementation & Results

- use SCIP (3.2.0) via its C++ interace

- parallelization with OpenMP
  - individual time limit
  - individual feasibility bounds

	hickerson-16	hickerson-18	knapsack_11_60
dimension	9	10	12
simplex volume	9.83×10^7	4.17×10^{14}	2.8×10^{14}
bottom volume	8.10×10^5	3.86×10^7	2.02×10^7
volume used			
integer programs solved			
improvement factor			
old runtime			
new runtime			

SUN xFire 4450, 4 Intel Xeon X7460 processors, 20 threads, $\mathrm{SCIPBound} = 10^6$

Implementation & Results

- use SCIP (3.2.0) via its C++ interace
- parallelization with OpenMP
  - individual time limit
  - individual feasibility bounds

	hickerson-16	hickerson-18	knapsack_11_60
dimension	9	10	12
simplex volume	9.83×10^7	4.17×10^{14}	2.8×10^{14}
bottom volume	8.10×10^5	3.86×10^7	2.02×10^7
volume used	3.93×10^6	5.47×10^7	2.39×10^7
integer programs solved			
improvement factor			
old runtime			
new runtime			

SUN xFire 4450, 4 Intel Xeon X7460 processors, 20 threads, $\mathrm{SCIPBound} = 10^6$

Implementation & Results

- use SCIP (3.2.0) via its C++ interace
- parallelization with OpenMP
  - individual time limit
  - individual feasibility bounds

	hickerson-16	hickerson-18	knapsack_11_60
dimension	9	10	12
simplex volume	9.83×10^7	4.17×10^{14}	2.8×10^{14}
bottom volume	8.10×10^5	3.86×10^7	2.02×10^7
volume used	3.93×10^6	5.47×10^7	2.39×10^7
integer programs solved	4	582016	11621
improvement factor			
old runtime			
new runtime			

SUN xFire 4450, 4 Intel Xeon X7460 processors, 20 threads, $\mathrm{SCIPBound} = 10^6$

Implementation & Results

- use SCIP (3.2.0) via its C++ interace
- parallelization with OpenMP
  - individual time limit
  - individual feasibility bounds

	hickerson-16	hickerson-18	knapsack_11_60
dimension	9	10	12
simplex volume	9.83×10^7	4.17×10^{14}	2.8×10^{14}
bottom volume	8.10×10^5	3.86×10^7	2.02×10^7
volume used	3.93×10^6	5.47×10^7	2.39×10^7
integer programs solved	4	582016	11621
improvement factor	25	7.62×10^6	1.17×10^7
old runtime			
new runtime			

SUN xFire 4450, 4 Intel Xeon X7460 processors, 20 threads, $\mathrm{SCIPBound} = 10^6$

Implementation & Results

- use SCIP (3.2.0) via its C++ interace
- parallelization with OpenMP
  - individual time limit
  - individual feasibility bounds

	hickerson-16	hickerson-18	knapsack_11_60
dimension	9	10	12
simplex volume	9.83×10^7	4.17×10^{14}	2.8×10^{14}
bottom volume	8.10×10^5	3.86×10^7	2.02×10^7
volume used	3.93×10^6	5.47×10^7	2.39×10^7
integer programs solved	4	582016	11621
improvement factor	25	7.62×10^6	1.17×10^7
old runtime	2s	> 12d	> 8d
new runtime			

SUN xFire 4450, 4 Intel Xeon X7460 processors, 20 threads, $\mathrm{SCIPBound} = 10^6$

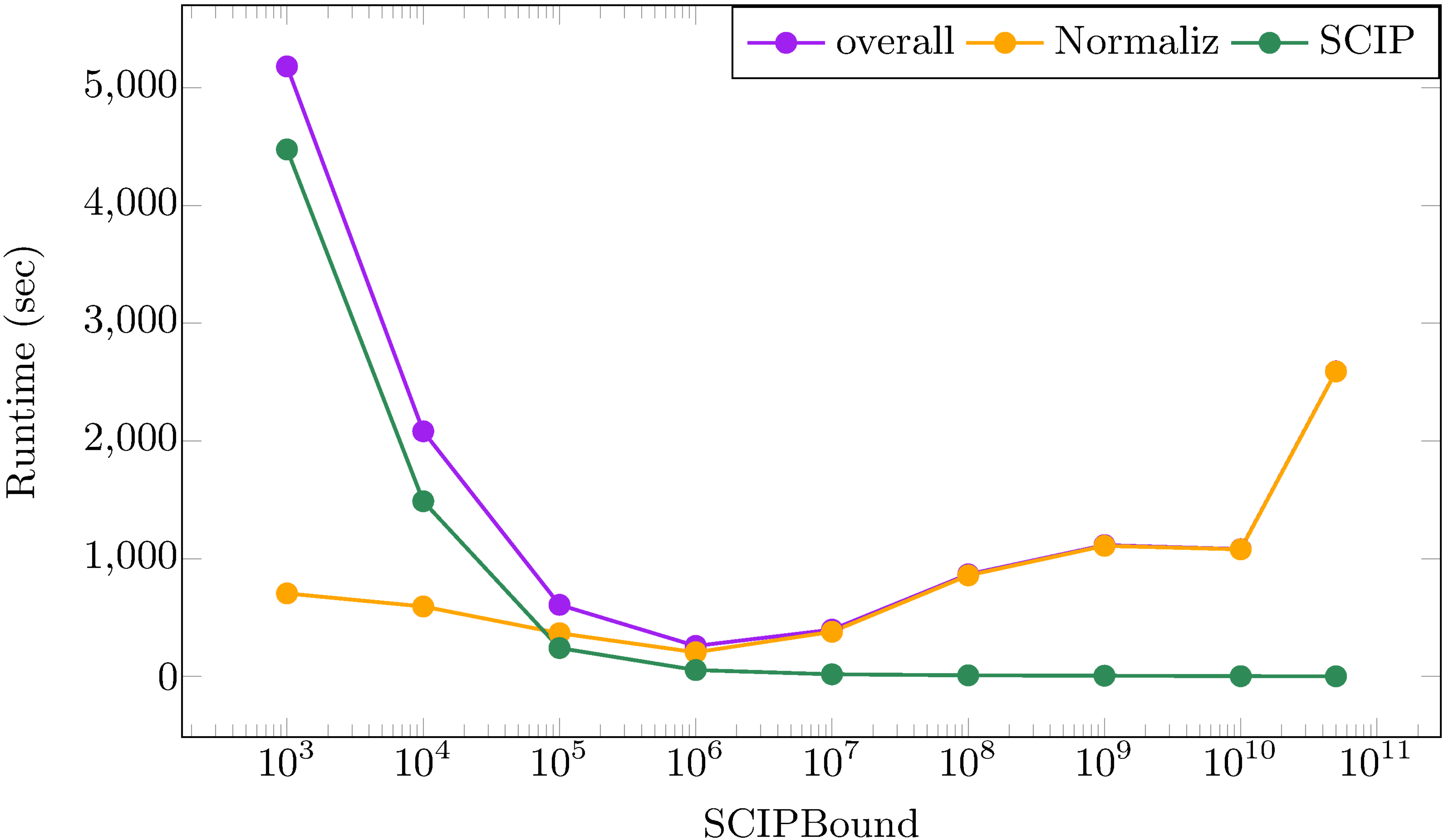
Implementation & Results

- use SCIP (3.2.0) via its C++ interace
- parallelization with OpenMP
  - individual time limit
  - individual feasibility bounds

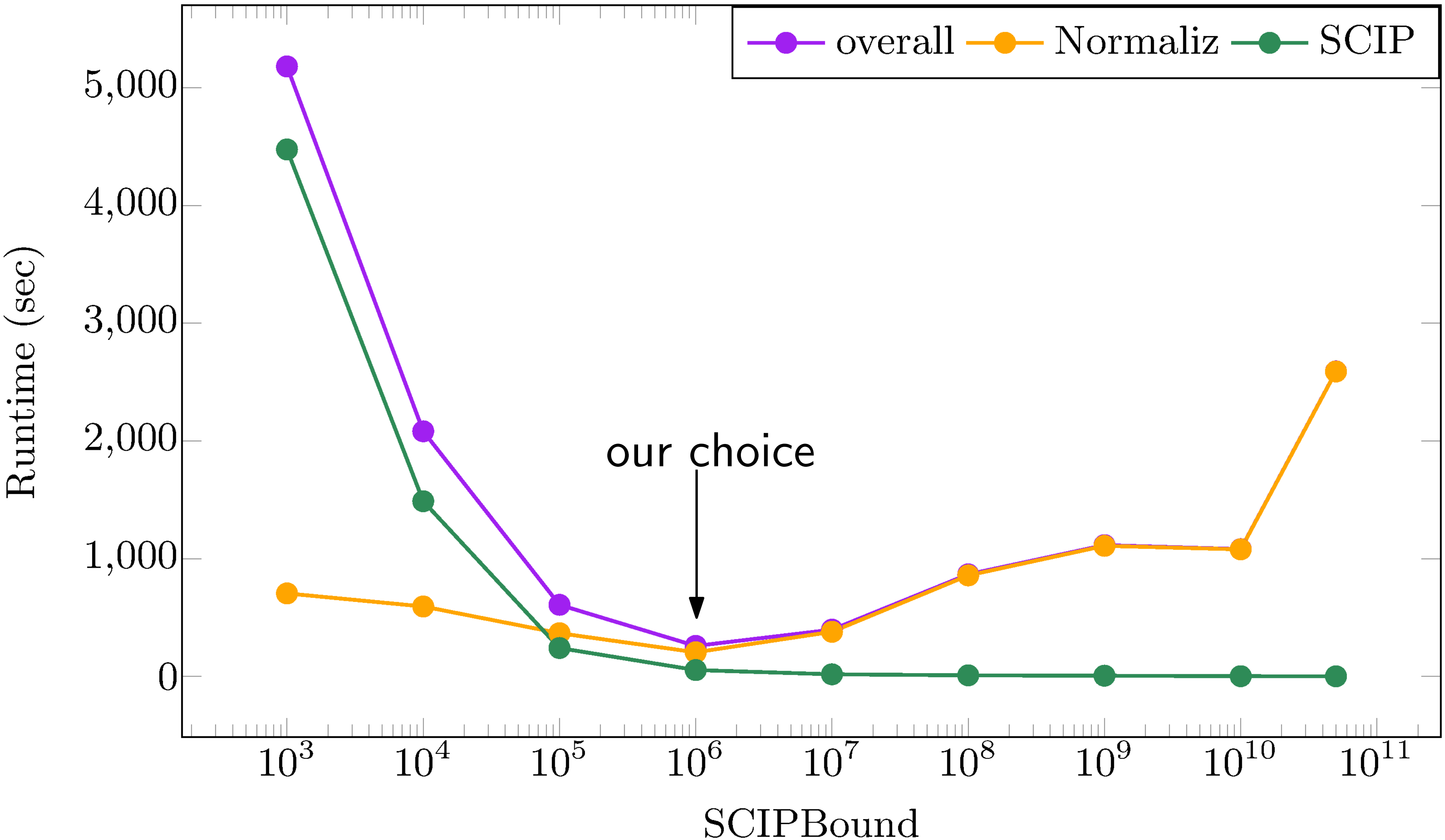
	hickerson-16	hickerson-18	knapsack_11_60
dimension	9	10	12
simplex volume	9.83×10^7	4.17×10^{14}	2.8×10^{14}
bottom volume	8.10×10^5	3.86×10^7	2.02×10^7
volume used	3.93×10^6	5.47×10^7	2.39×10^7
integer programs solved	4	582016	11621
improvement factor	25	7.62×10^6	1.17×10^7
old runtime	2s	> 12d	> 8d
new runtime	0.5s	46s	5.1s

SUN xFire 4450, 4 Intel Xeon X7460 processors, 20 threads, $\mathrm{SCIPBound} = 10^6$

Implementation & Results



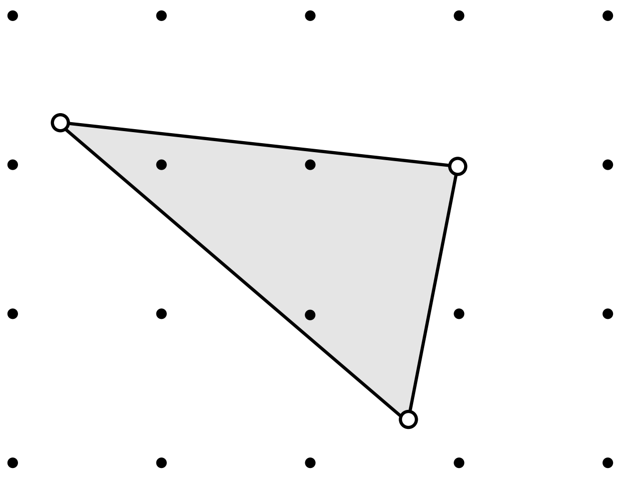
Implementation & Results



Approximation

The Algorithm

1. Look at the cross section at level 1 of the (transformed) simplex.

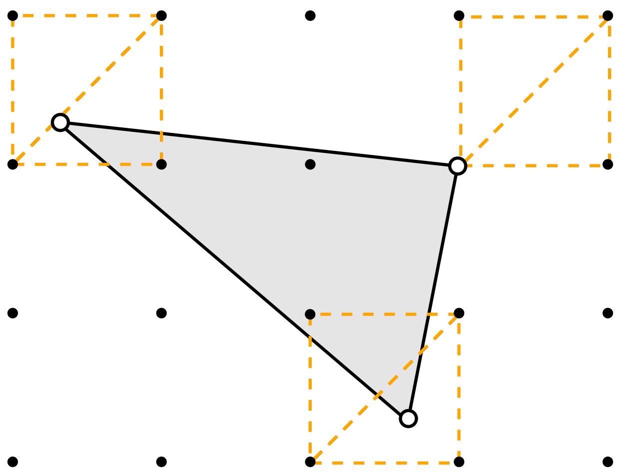


cross section at level 1

The Algorithm

1. Look at the cross section at level 1 of the (transformed) simplex.

2. For each extreme ray/point, triangulate the lattice cube around it using the hyperplane arrangement $\mathcal{A}_n = \{x_i = x_j\}$.



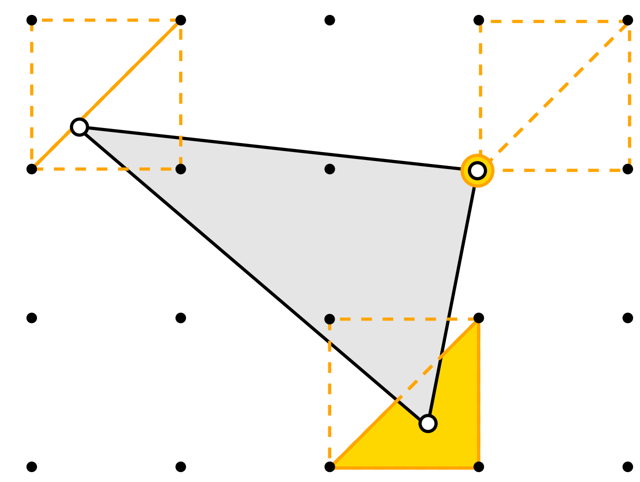
cross section at level 1

The Algorithm

1. Look at the cross section at level 1 of the (transformed) simplex.

2. For each extreme ray/point, triangulate the lattice cube around it using the hyperplane arrangement $\mathcal{A}_n = \{x_i = x_j\}$.

3. Detect the minimal face containing the point and collect its vertices (at most d).



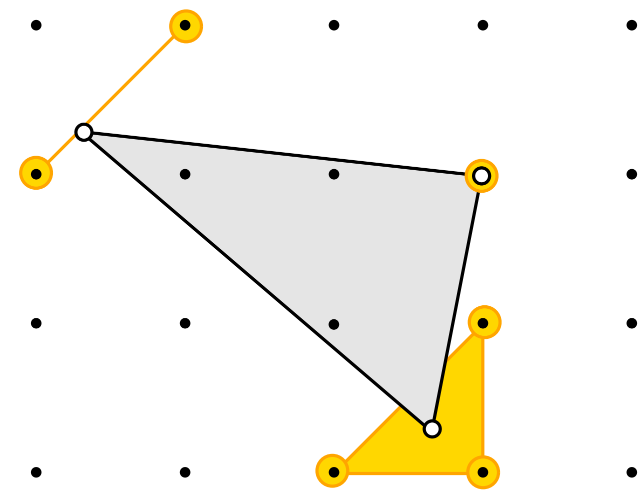
cross section at level 1

The Algorithm

1. Look at the cross section at level 1 of the (transformed) simplex.

2. For each extreme ray/point, triangulate the lattice cube around it using the hyperplane arrangement $\mathcal{A}_n = \{x_i = x_j\}$.

3. Detect the minimal face containing the point and collect its vertices (at most d).



cross section at level 1

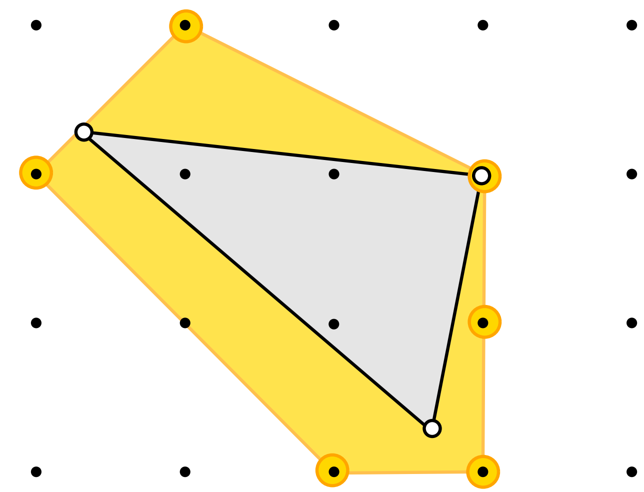
The Algorithm

1. Look at the cross section at level 1 of the (transformed) simplex.

2. For each extreme ray/point, triangulate the lattice cube around it using the hyperplane arrangement $\mathcal{A}_n = \{x_i = x_j\}$.

3. Detect the minimal face containing the point and collect its vertices (at most d).

4. Create a candidate list of the new cone, intersect it with the original cone and do local reduction.



cross section at level 1

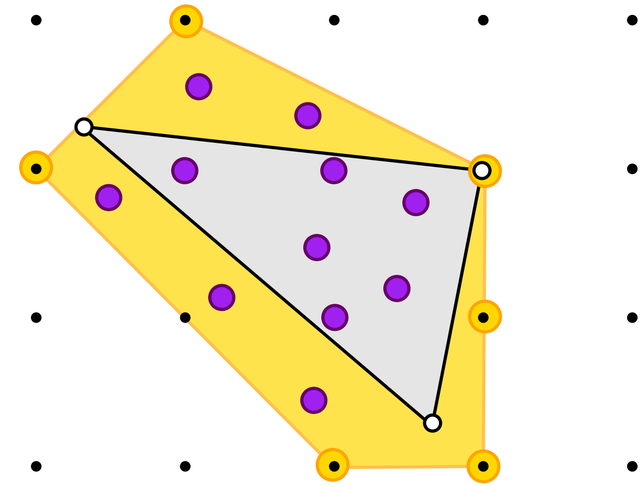
The Algorithm

1. Look at the cross section at level 1 of the (transformed) simplex.

2. For each extreme ray/point, triangulate the lattice cube around it using the hyperplane arrangement $\mathcal{A}_n = \{x_i = x_j\}$.

3. Detect the minimal face containing the point and collect its vertices (at most d).

4. Create a candidate list of the new cone, intersect it with the original cone and do local reduction.



cross section at level 1

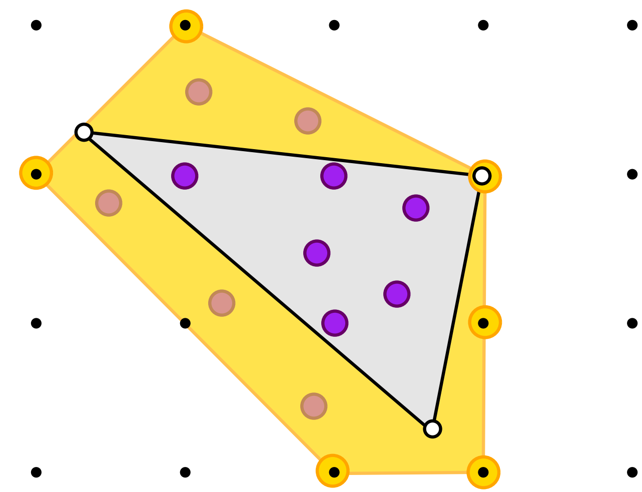
The Algorithm

1. Look at the cross section at level 1 of the (transformed) simplex.

2. For each extreme ray/point, triangulate the lattice cube around it using the hyperplane arrangement $\mathcal{A}_n = \{x_i = x_j\}$.

3. Detect the minimal face containing the point and collect its vertices (at most d).

4. Create a candidate list of the new cone, intersect it with the original cone and do local reduction.



cross section at level 1

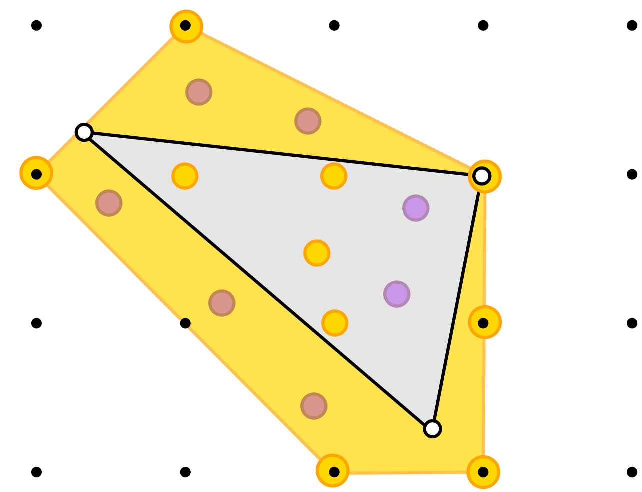
The Algorithm

1. Look at the cross section at level 1 of the (transformed) simplex.

2. For each extreme ray/point, triangulate the lattice cube around it using the hyperplane arrangement $\mathcal{A}_n = \{x_i = x_j\}$.

3. Detect the minimal face containing the point and collect its vertices (at most d).

4. Create a candidate list of the new cone, intersect it with the original cone and do local reduction.



cross section at level 1

The Algorithm

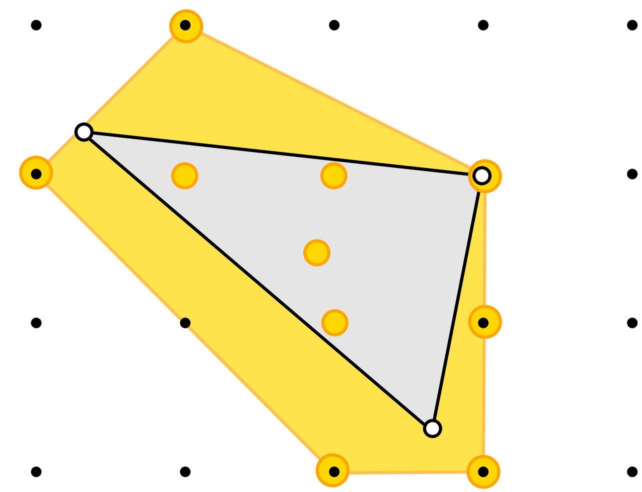
1. Look at the cross section at level 1 of the (transformed) simplex.

2. For each extreme ray/point, triangulate the lattice cube around it using the hyperplane arrangement $\mathcal{A}_n = \{x_i = x_j\}$.

3. Detect the minimal face containing the point and collect its vertices (at most d).

4. Create a candidate list of the new cone, intersect it with the original cone and do local reduction.

$\Rightarrow$ list of points $\mathcal{B}$ (bottom candidates)



cross section at level 1

The Algorithm

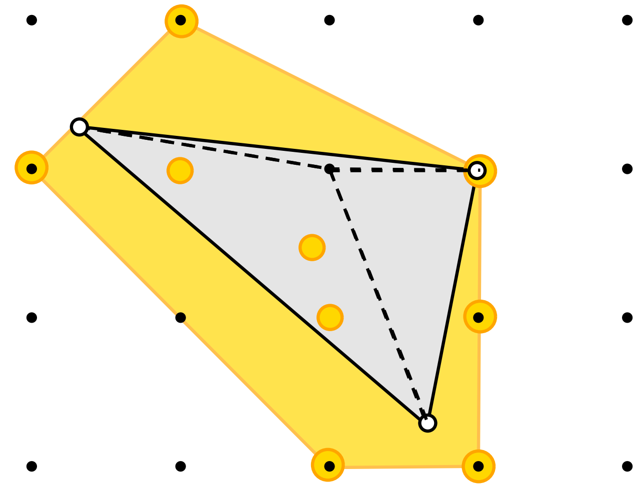
1. Look at the cross section at level 1 of the (transformed) simplex.

2. For each extreme ray/point, triangulate the lattice cube around it using the hyperplane arrangement $\mathcal{A}_n = \{x_i = x_j\}$.

3. Detect the minimal face containing the point and collect its vertices (at most d).

4. Create a candidate list of the new cone, intersect it with the original cone and do local reduction.

$\Rightarrow$ list of points $\mathcal{B}$ (bottom candidates)



cross section at level 1

Choose a grading minimizing point from $\mathcal{B}$ and continue as before.

The Algorithm

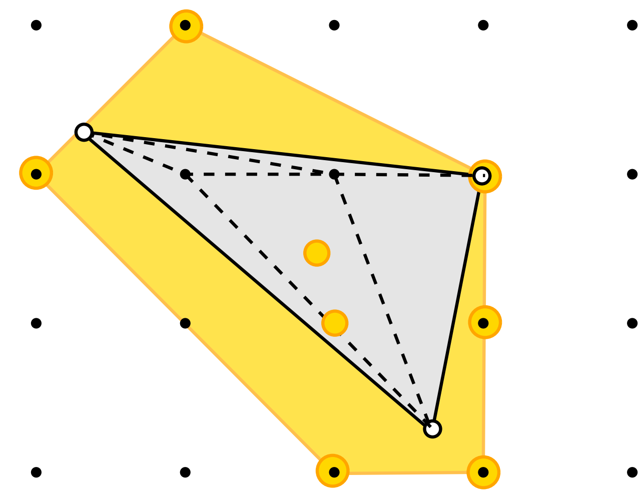
1. Look at the cross section at level 1 of the (transformed) simplex.

2. For each extreme ray/point, triangulate the lattice cube around it using the hyperplane arrangement $\mathcal{A}_n = \{x_i = x_j\}$.

3. Detect the minimal face containing the point and collect its vertices (at most d).

4. Create a candidate list of the new cone, intersect it with the original cone and do local reduction.

$\Rightarrow$ list of points $\mathcal{B}$ (bottom candidates)



cross section at level 1

Choose a grading minimizing point from $\mathcal{B}$ and continue as before.

Results

	hickerson-16		hickerson-18		knapsack_11_60	
simplex vol	$9.83\,\mathrm{e}\,7$		$4.17\,\mathrm{e}\,14$		$2.8\,\mathrm{e}\,14$	
bottom vol	$8.10\,\mathrm{e}\,5$		$3.86\,\mathrm{e}\,7$		$2.02\,\mathrm{e}\,7$	
	(1)	(2)	(1)	(2)	(1)	(2)
our vol	$3.93\,\mathrm{e}\,6$	$3.93\,\mathrm{e}\,6$	$5.47\,\mathrm{e}\,7$	$8.42\,\mathrm{e}\,7$	$2.39\,\mathrm{e}\,7$	$9.36\,\mathrm{e}\,9$
factor	25	25	$7.62\,\mathrm{e}\,6$	$4.95\,\mathrm{e}\,6$	$1.09\,\mathrm{e}\,7$	$2.99\,\mathrm{e}\,4$
old time	2s		$>$12d		$>$8d	
new time	0.5s	0.4s	46s	50s	5s	2m30s

Improvements & Outlook

⋆ If "large" simplices are remaining (both cases): approximate on a higher level.
(Which one?)

Improvements & Outlook

- If "large" simplices are remaining (both cases): approximate on a higher level.
  (Which one?)

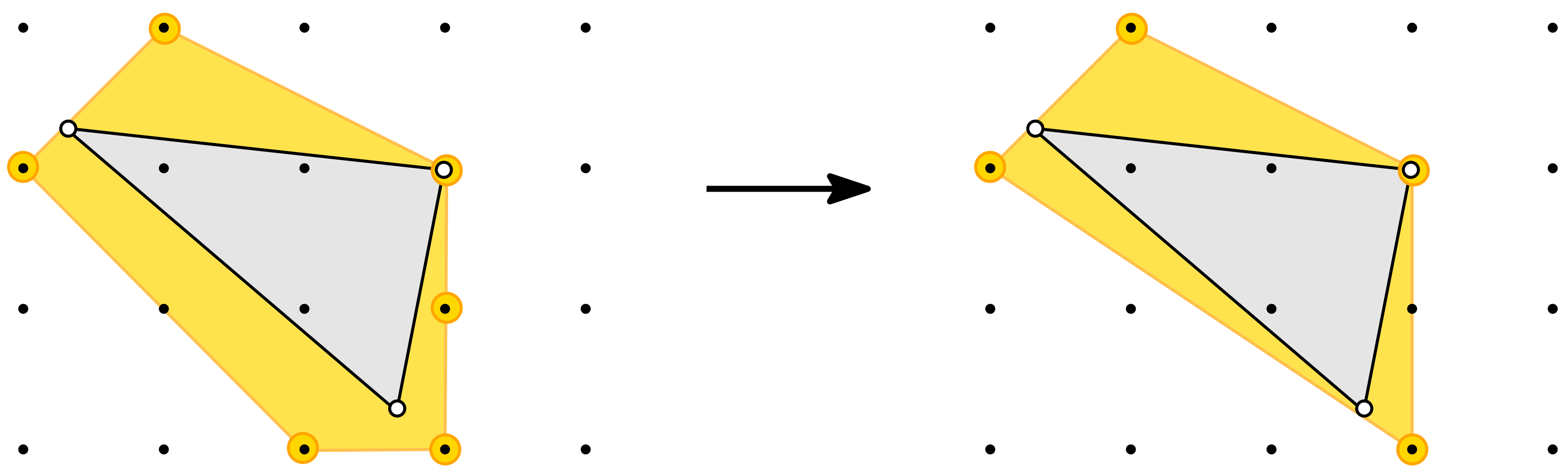
- Tweak settings in SCIP (time bounds etc.).

Improvements & Outlook

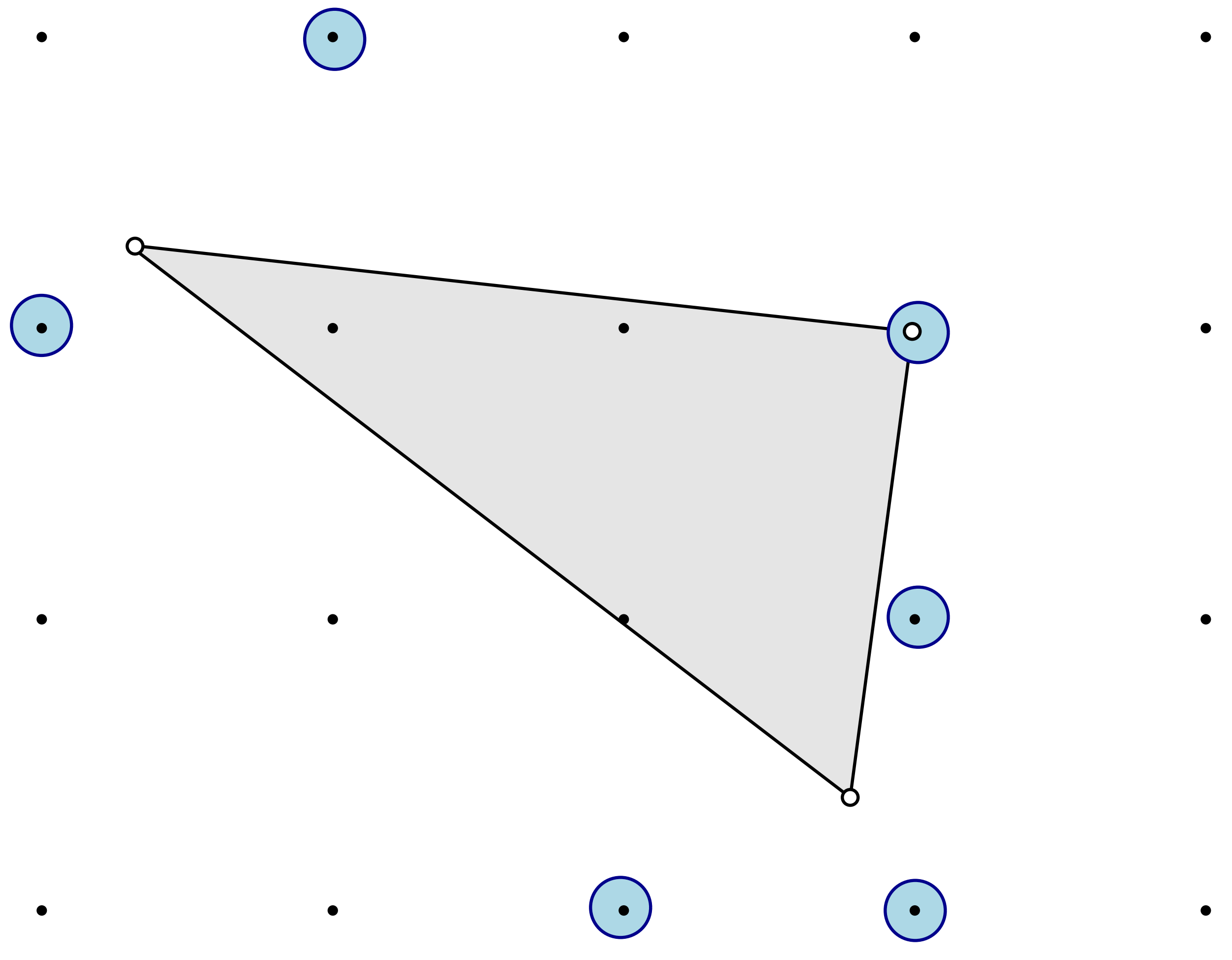
- If "large" simplices are remaining (both cases): approximate on a higher level.
  (Which one?)

- Tweak settings in SCIP (time bounds etc.).

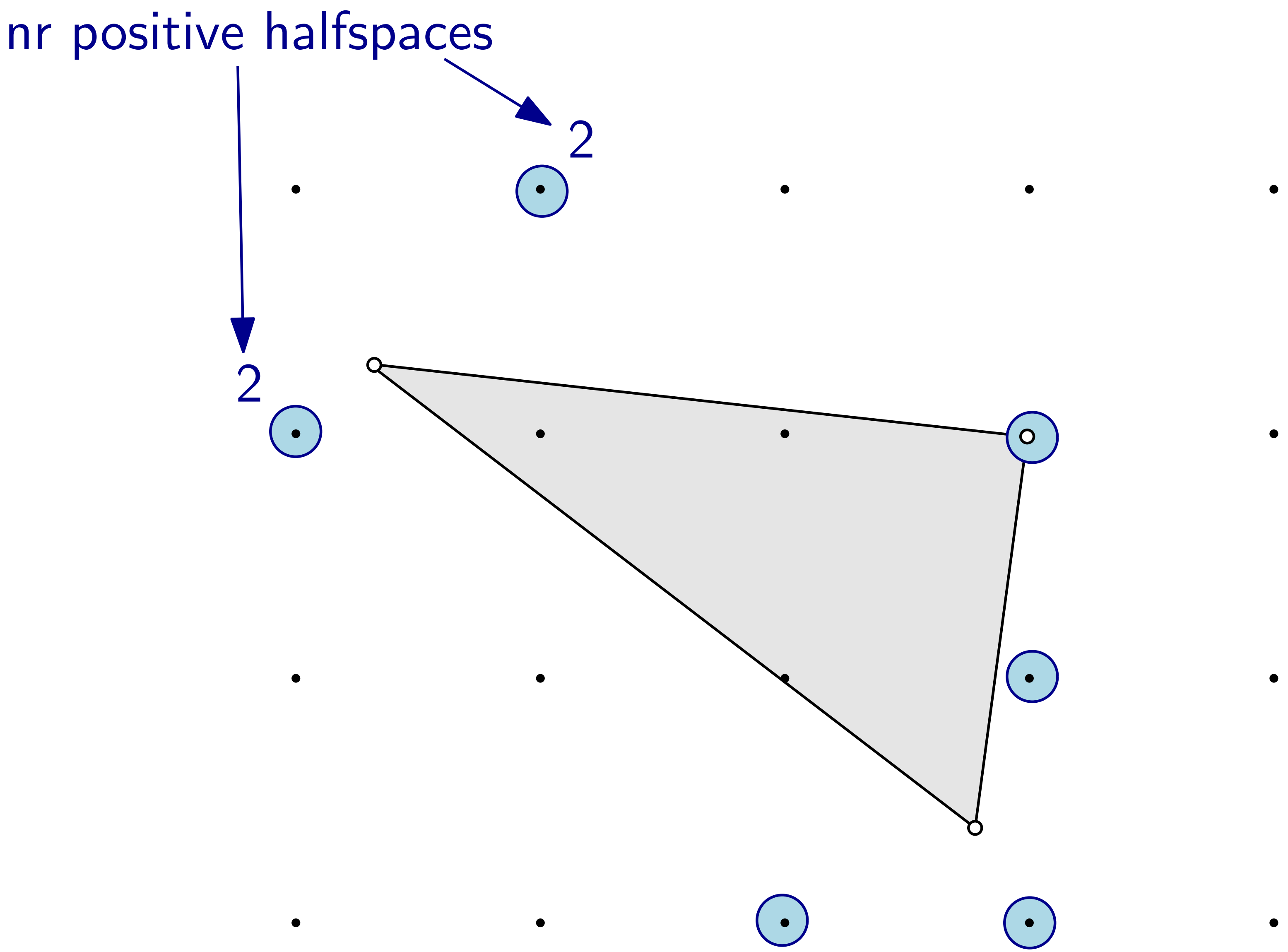
- Use less generators for approximating cone.
  (Partial Fourier-Motzkin Elimination)



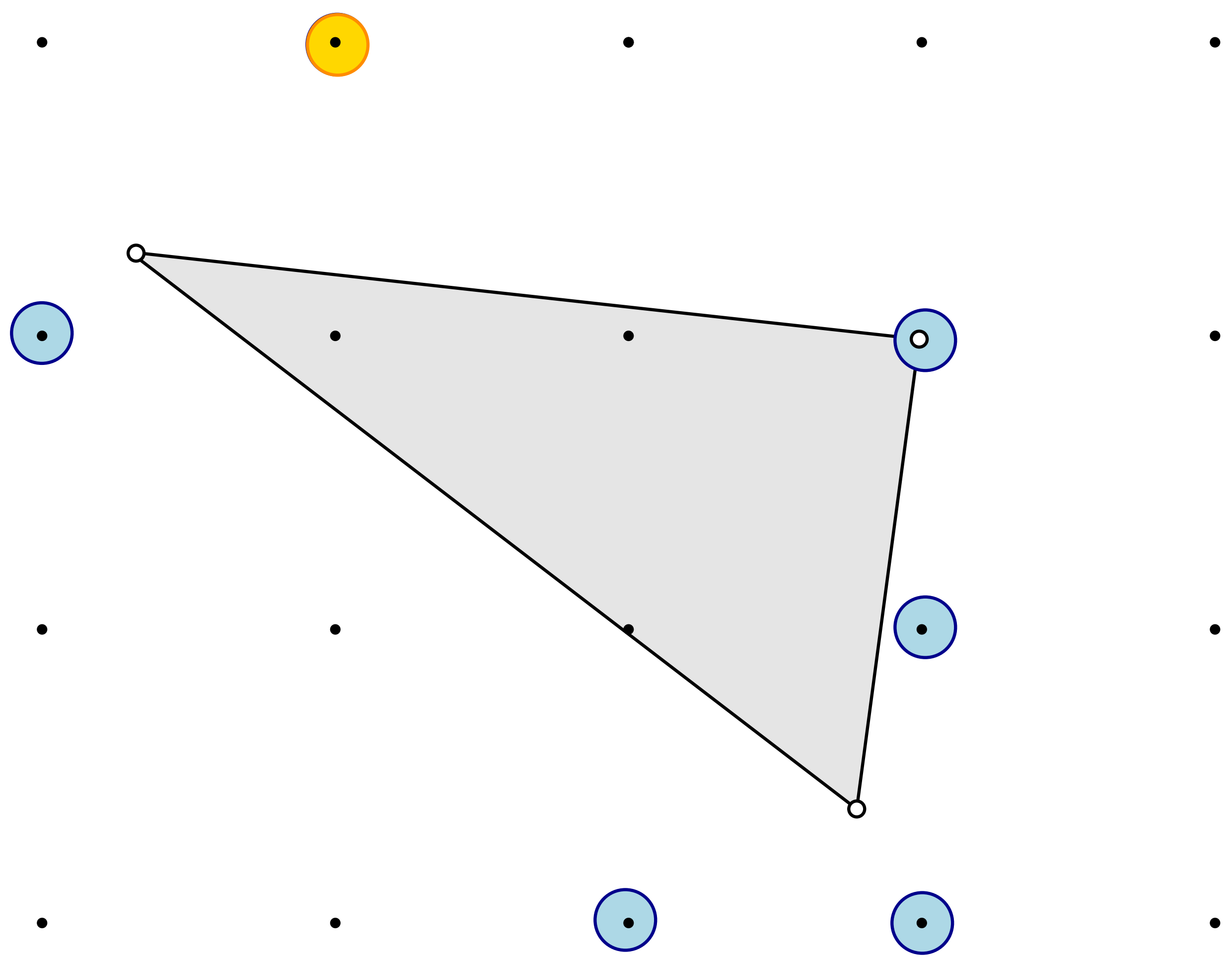
Partial Fourier-Motzkin Elimination



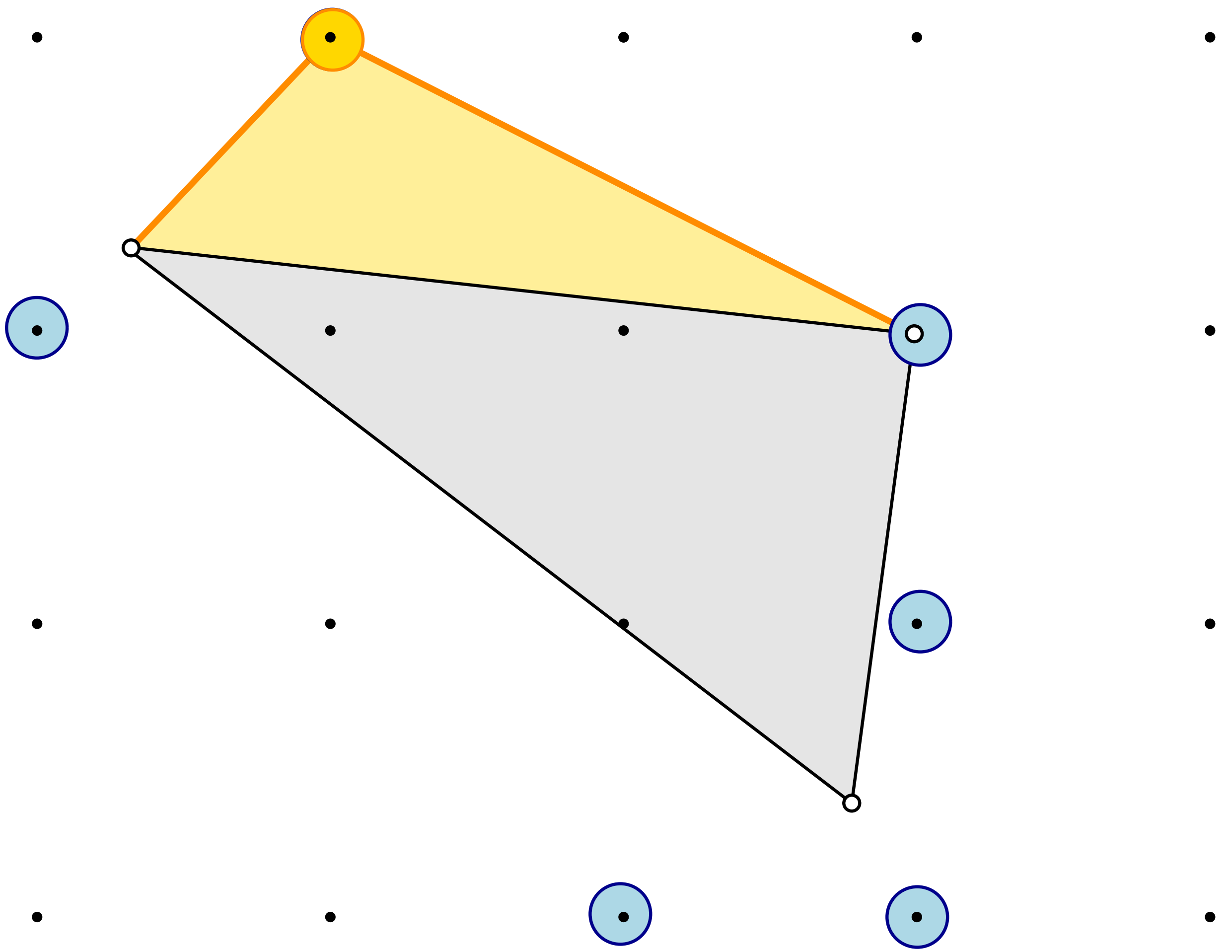
Partial Fourier-Motzkin Elimination



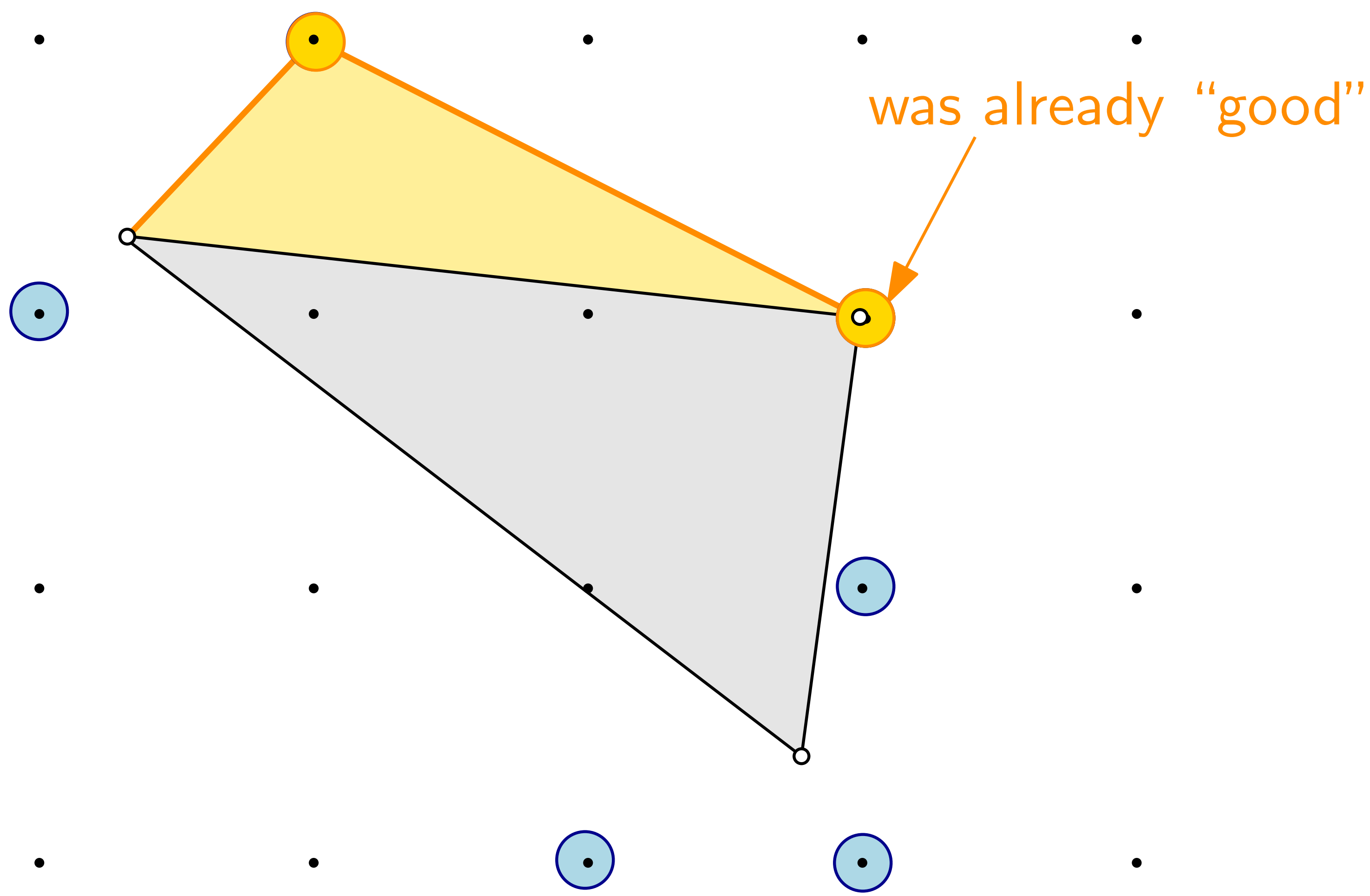
Partial Fourier-Motzkin Elimination



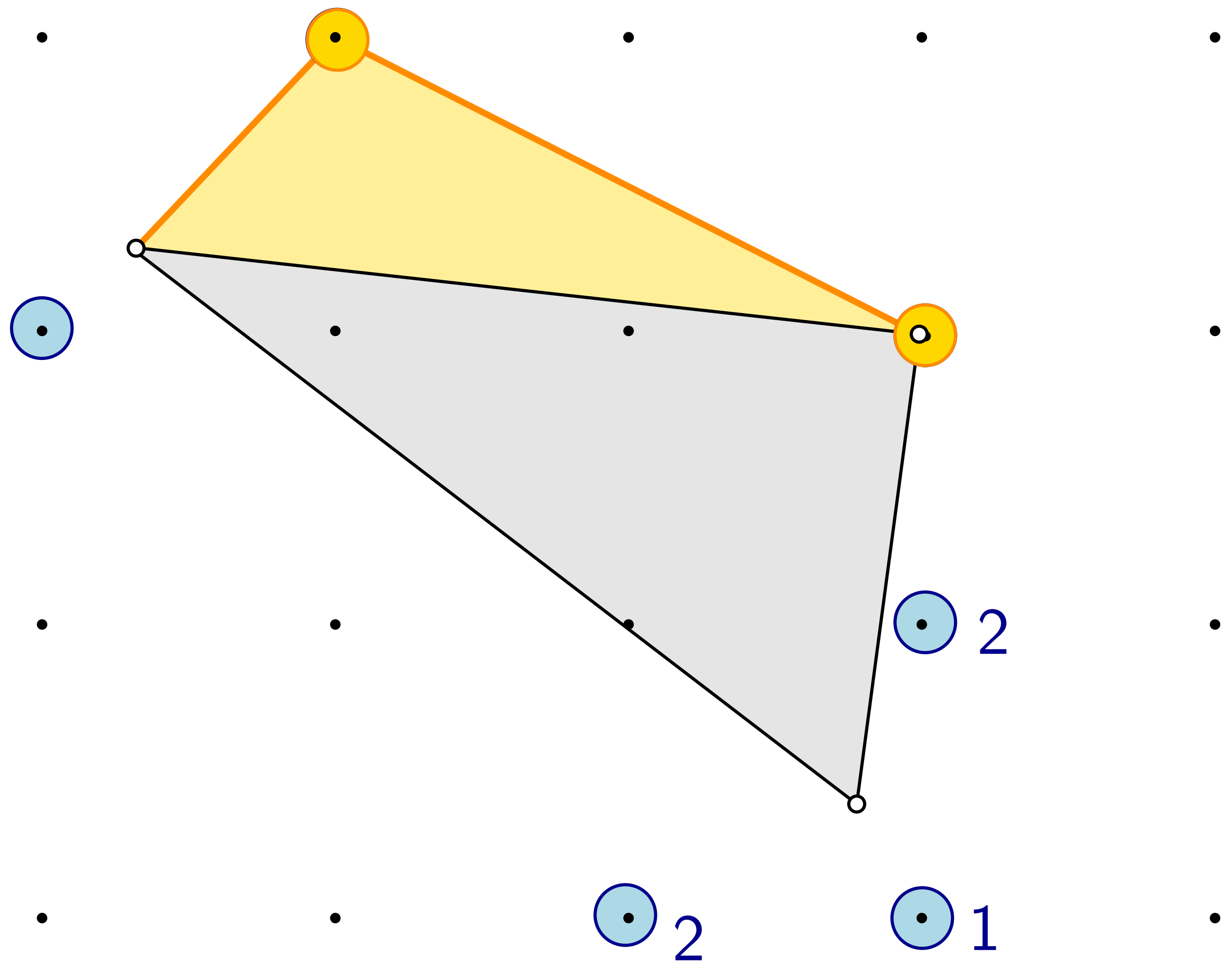
Partial Fourier-Motzkin Elimination



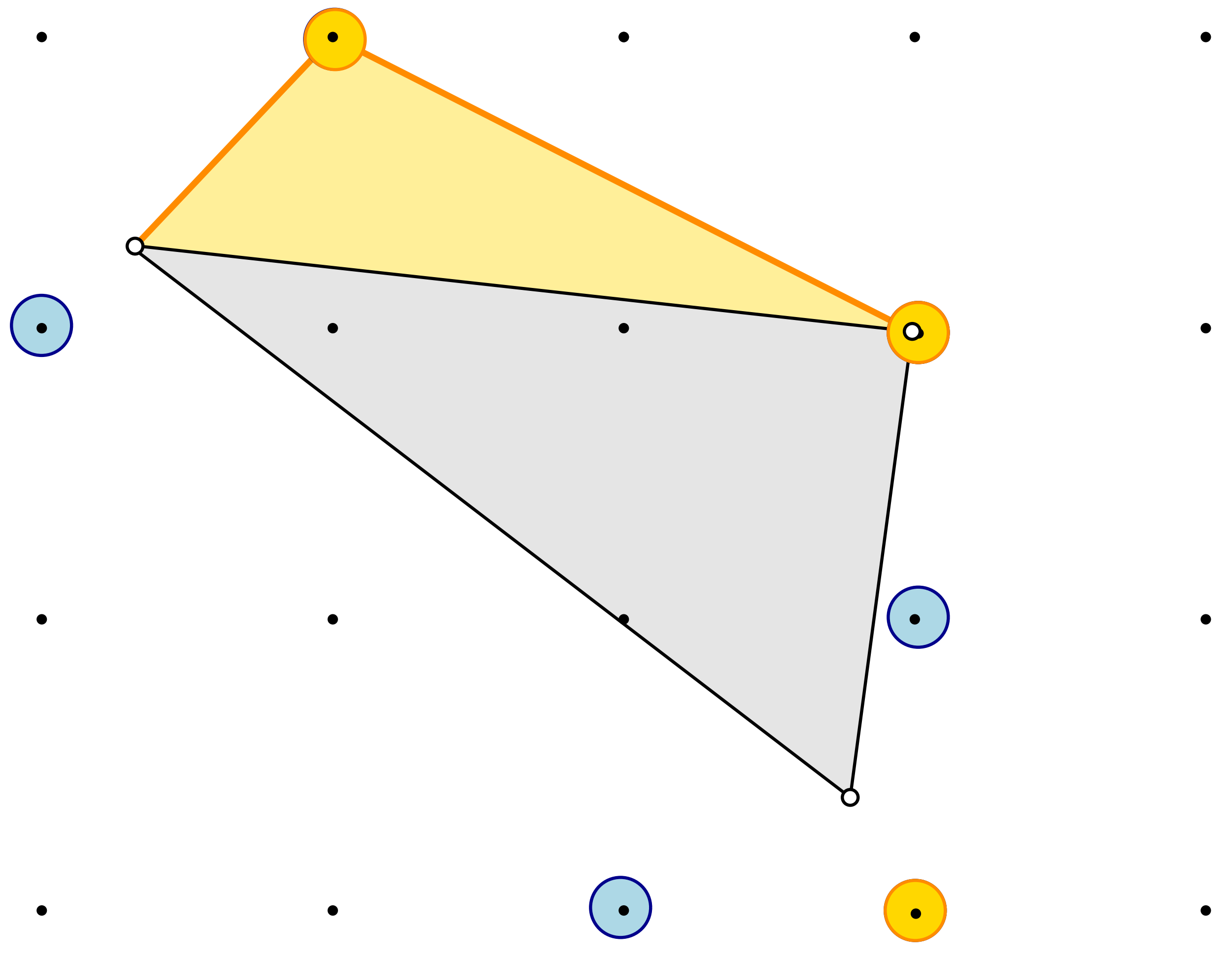
Partial Fourier-Motzkin Elimination



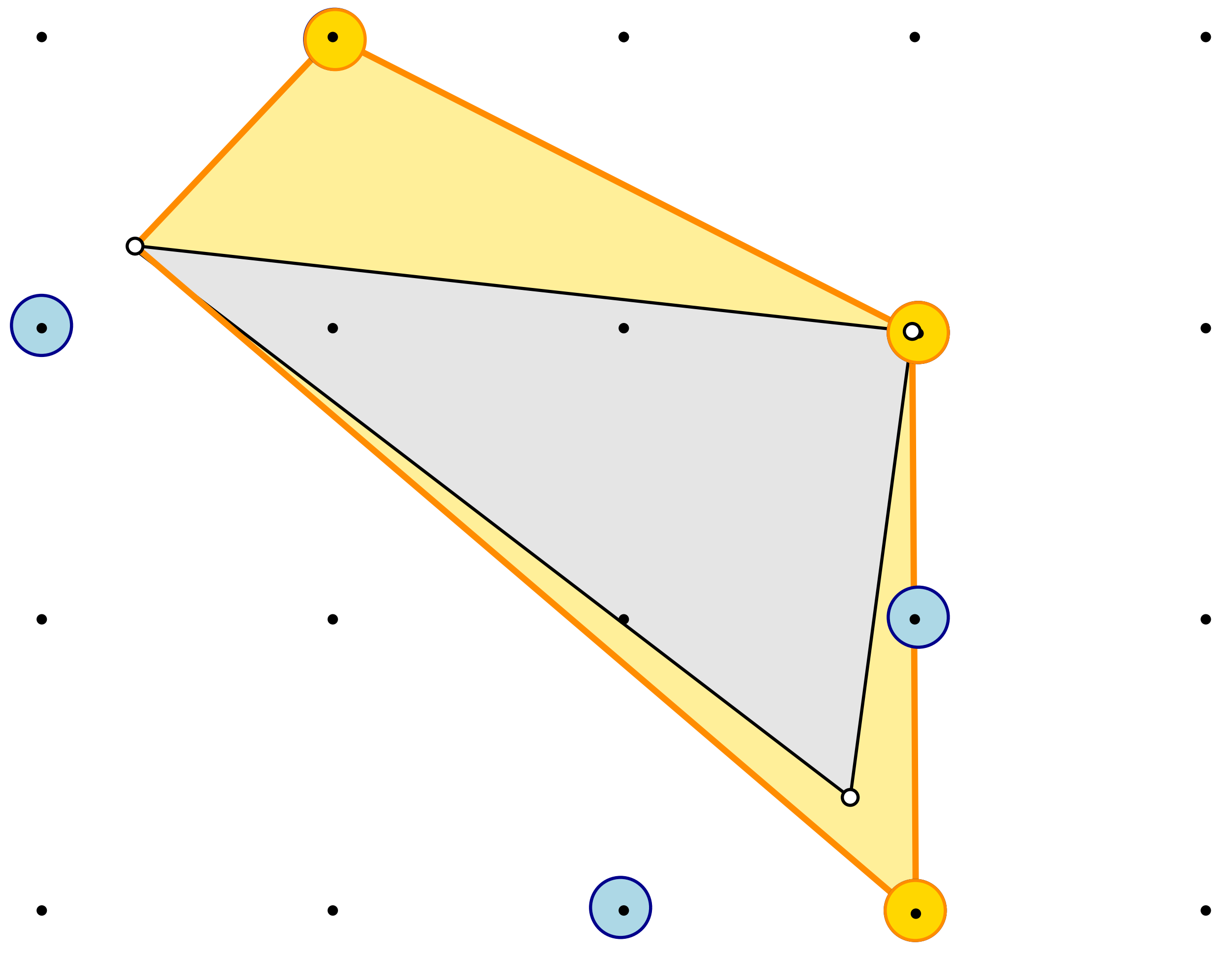
Partial Fourier-Motzkin Elimination



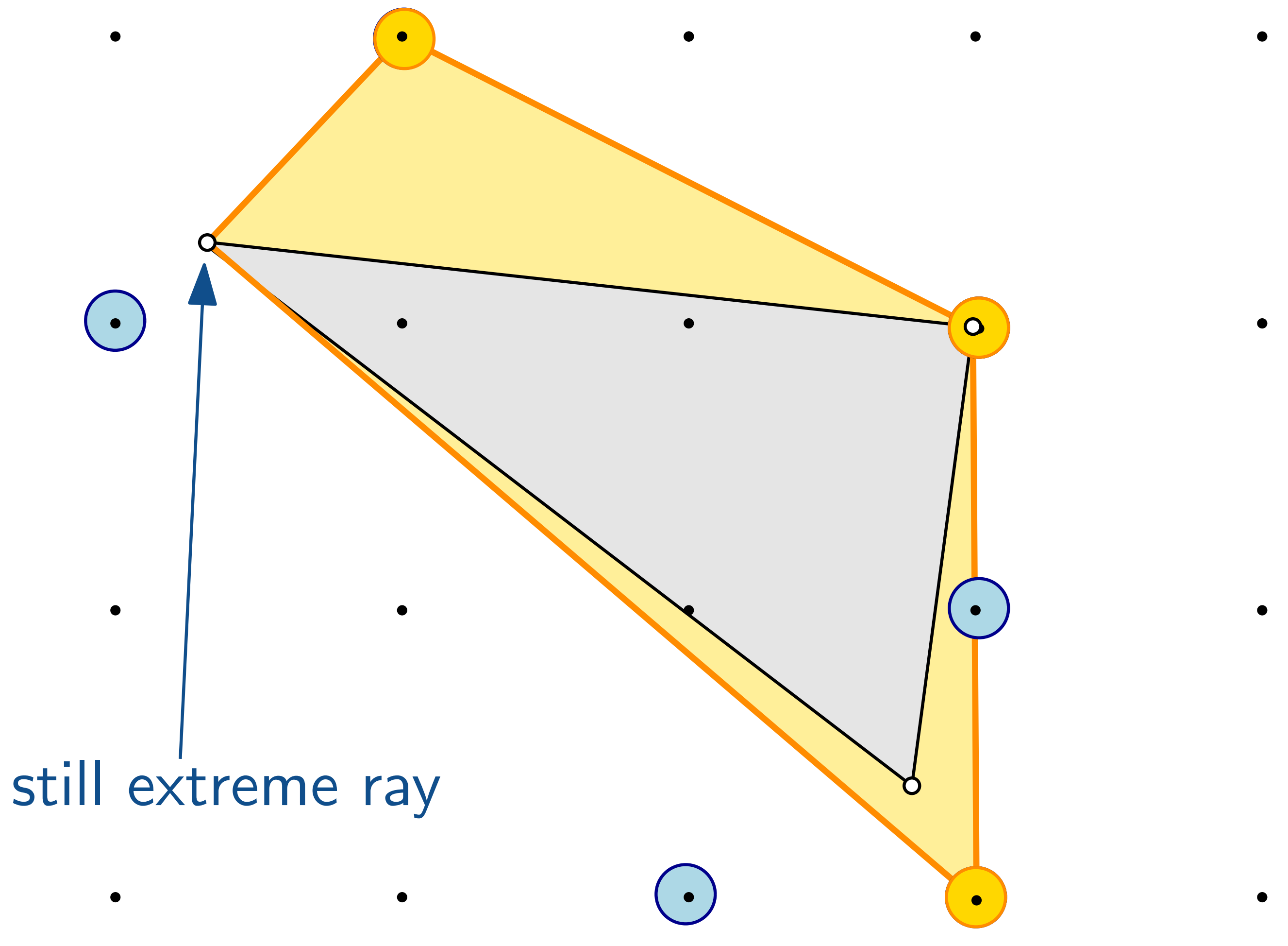
Partial Fourier-Motzkin Elimination



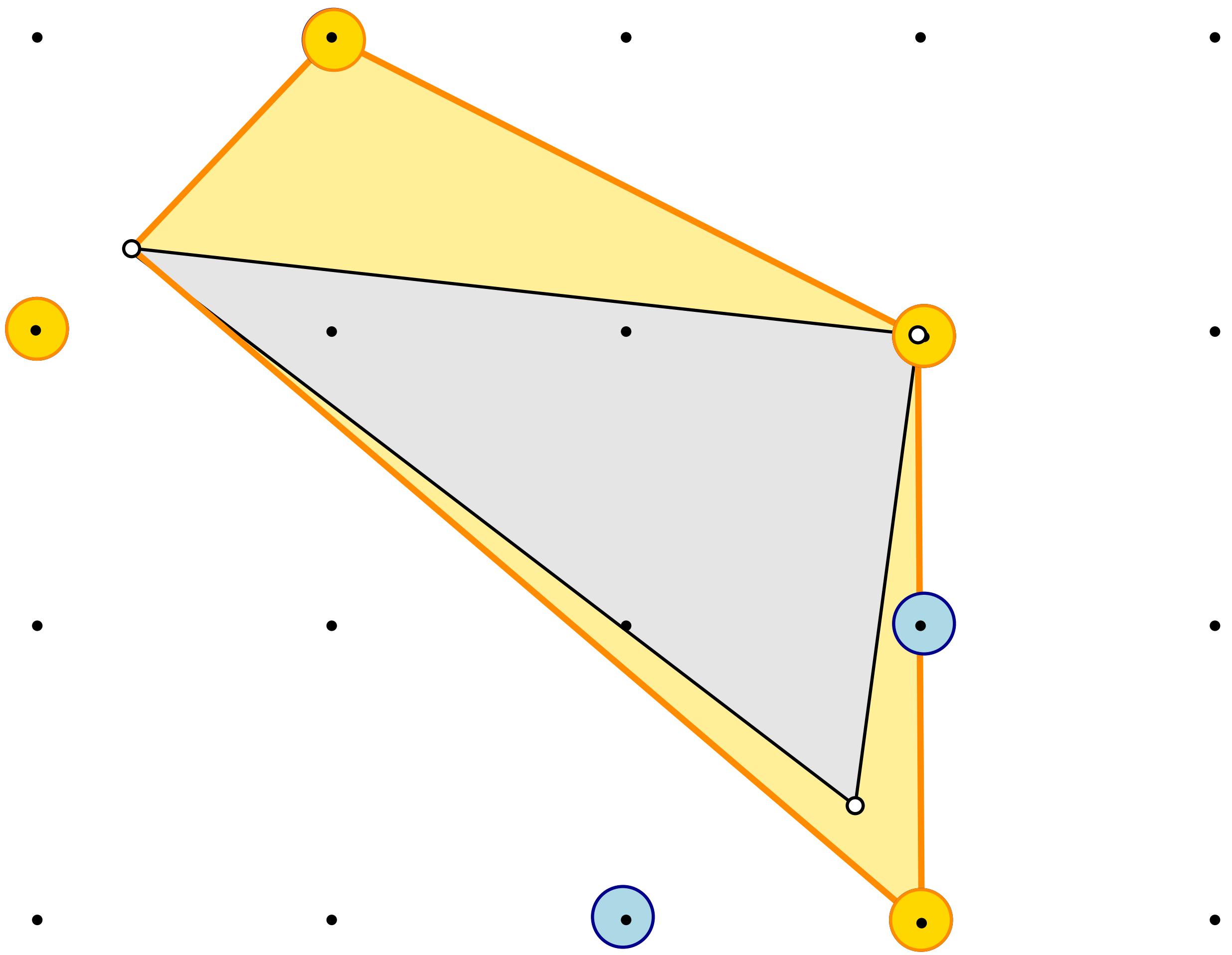
Partial Fourier-Motzkin Elimination



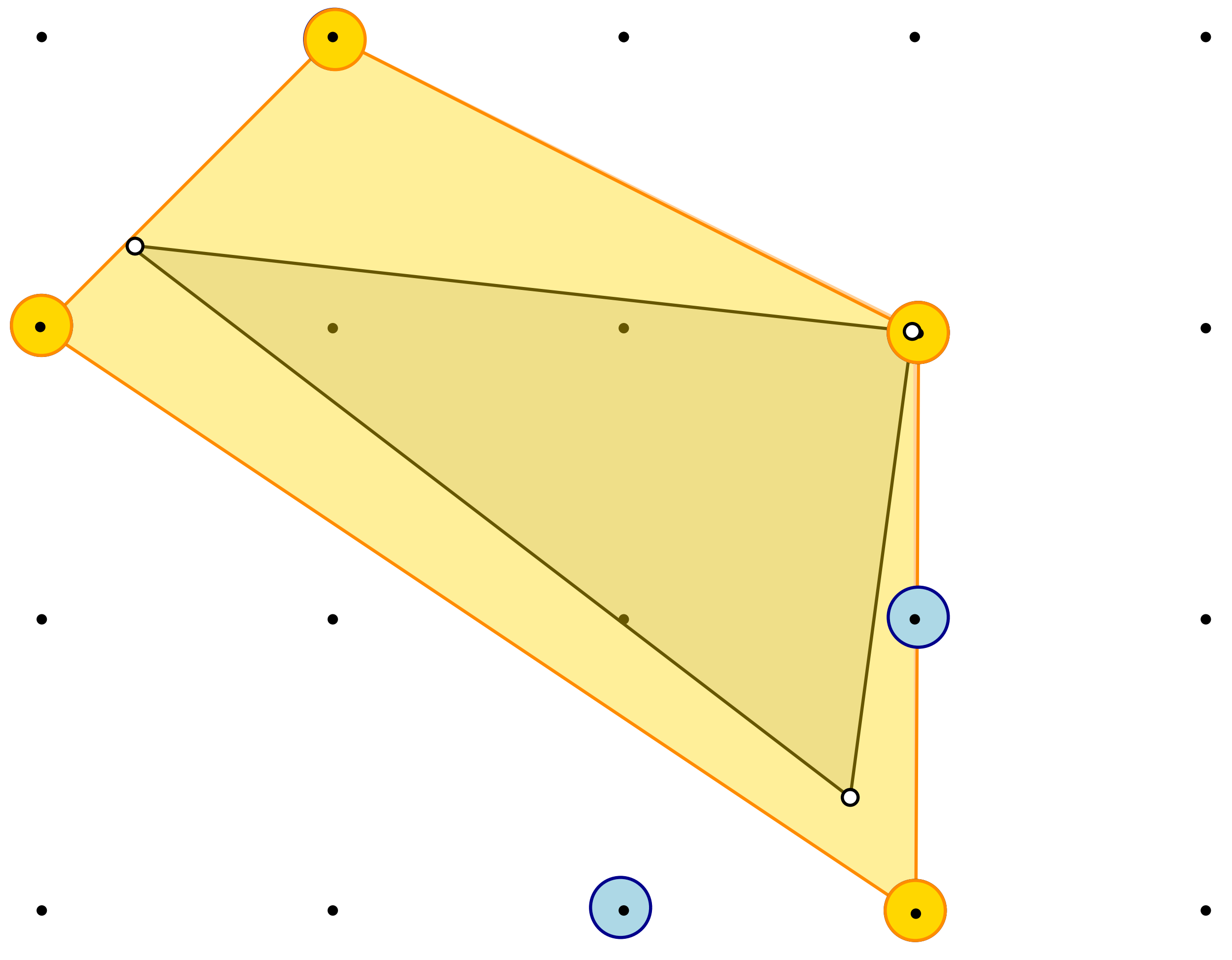
Partial Fourier-Motzkin Elimination



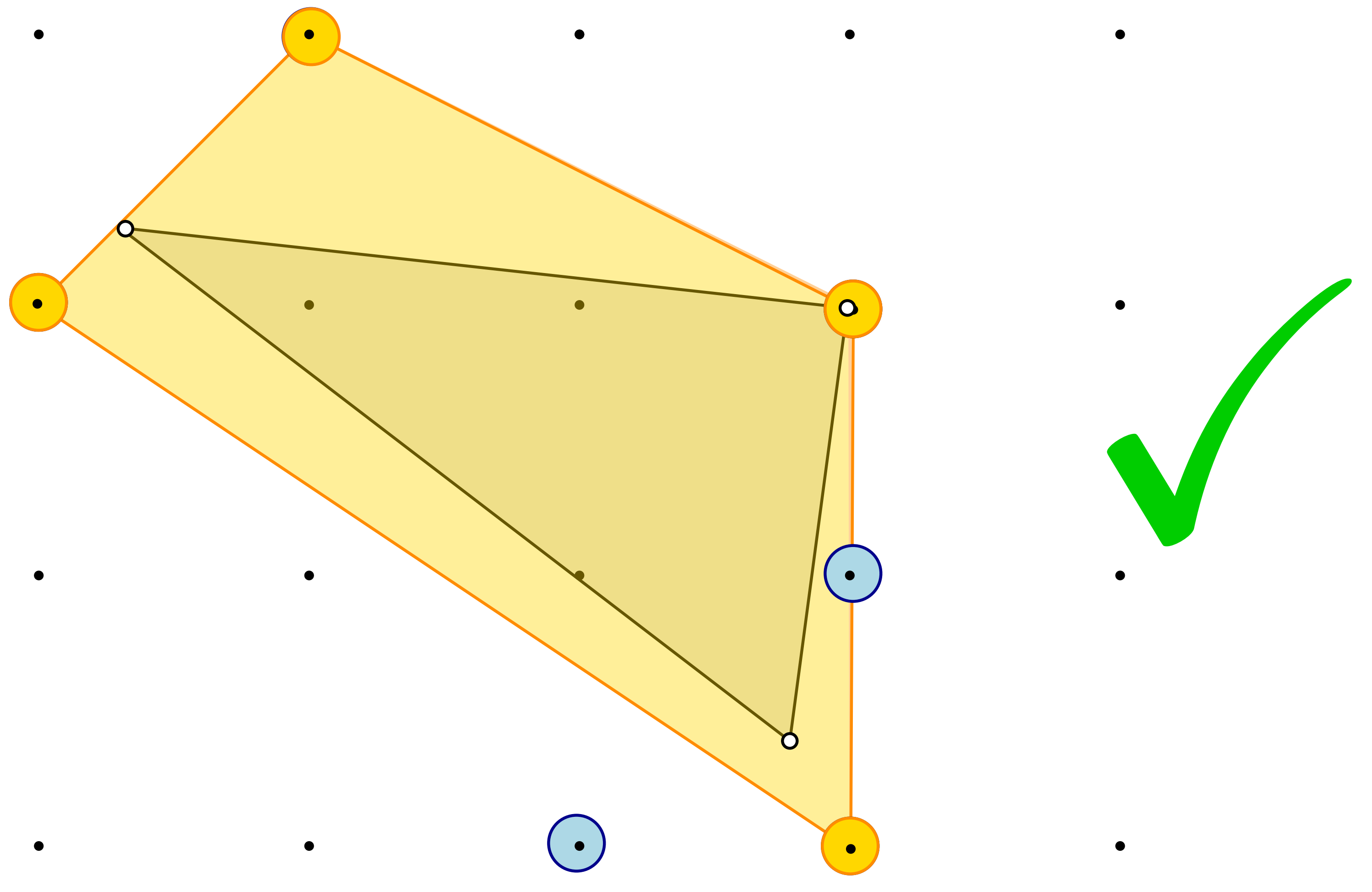
Partial Fourier-Motzkin Elimination



Partial Fourier-Motzkin Elimination

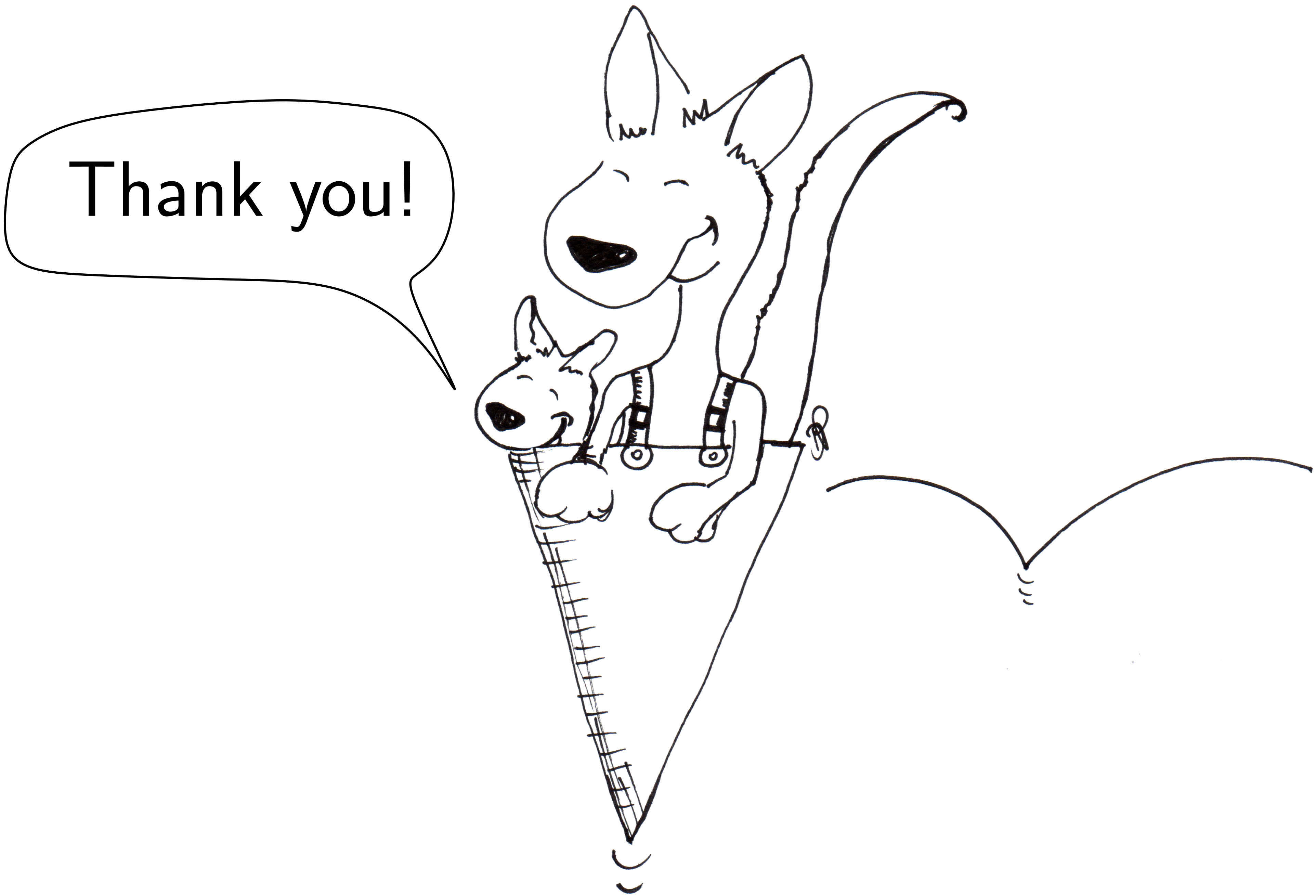


Partial Fourier-Motzkin Elimination



Demo

```
                                                    \.....|
                    Normaliz 3.0.0                   \....|
                                                      \...|
     (C) The Normaliz Team, University of Osnabrueck   \..|
                    September 2015                      \.|
                                                         \|
************************************************************
Compute: DefaultMode
Computing extreme rays as support hyperplanes of the dual cone:
************************************************************
starting primal algorithm (only support hyperplanes) ...
Generators sorted lexicographically
Start simplex 1 2 3 4 5 6 7 8 9 10
Checking for pointed ... done.
Select extreme rays via comparison ... done.
------------------------------------------------------------
************************************************************
starting primal algorithm with full triangulation ...
Roughness 7
Generators sorted by degree and lexicographically
Generators per degree:
2: 2  4: 2  14: 6
Start simplex 1 2 3 4 5 6 7 8 9 10
Pointed since graded
evaluating 1 simplices
||||||||||||||||||||||||||||||||||||||||||||||||||
1 simplices, 0 HB candidates accumulated.
1 large simplices stored
Evaluating 1 large simplices
Large simplex 1 / 1
simplex volume 416728074151872
**************************************************
```


Thank you!