Lattice-Based Memory Allocation

Gilles Villard

CNRS, Laboratoire LIP, ENS Lyon France

Joint work with **Alain Darte** (CNRS, LIP) and **Rob Schreiber** (HP Labs)

Int. Conf. Compilers, Architecture and Synthesis for Embedded Systems, ACM Press, Nov. 2003

Research Report LIP, May 2004, `http://www.ens-lyon.fr/LIP/Pub`

MOCAA, May 6, 2004, University of Waterloo

CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE

INRIA RHÔNE-ALPES

Introduction

Example 1.

do $i = 0,\ N-1$
  do $j = 0,\ N-1$
    S: $A(i,j) = \ldots$
  end
end

Schedule: $\theta(S,i,j) = Ni + j$

do $i = 0,\ N-1$
  do $j = 0,\ N-1$
    T: $B(i,j) = A(i,j) + \ldots$
  end
end

$\theta(S,i,j) = Ni + j + 1$
i.e., **one "clock-cycle" later**

Example 1.

do $i = 0,\ N-1$
  do $j = 0,\ N-1$
    S: $A(i,j) = \ldots$
  end
end

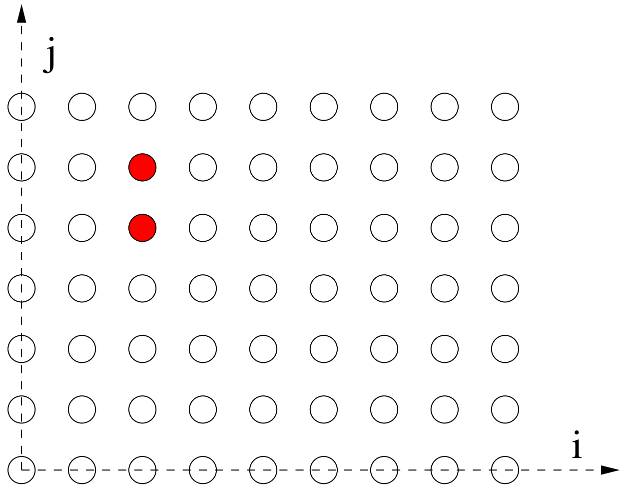
do $i = 0,\ N-1$
  do $j = 0,\ N-1$
    T: $B(i,j) = A(i,j) + \ldots$
  end
end

Schedule: $\theta(S,i,j) = Ni + j$

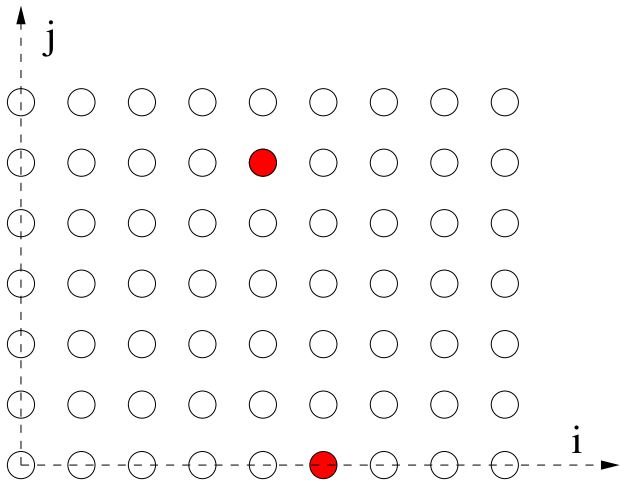
$\theta(S,i,j) = Ni + j + 1$
i.e., **one "clock-cycle" later**

Design **intermediate buffers** for A with **memory reuse**?

Some array values cannot share the same buffer location,
e.g. $A(i,j)$ and $A(i,j+1)$ since $A(i,j)$ is required later by the second loop,
we say that corresponding indices are **conflicting** (relation $\bowtie$):



$(i,j) \bowtie (i,j+1)$



$(i,N-1) \bowtie (i+1,0) \quad (N=6)$

The allocation function

$$\sigma : (i, j) \mapsto Ni + j \bmod 2,$$

which stores $A(i, j)$ in Buffer$[\sigma(i, j)]$, is a **valid allocation** (1D), indeed,

$(i, j) \bowtie (i, j+1)$: $Ni + j \neq Ni + j + 1 \bmod 2$

$(i, N-1) \bowtie (i+1, 0)$: $Ni + N - 1 \neq Ni + N \bmod 2$

Conflicting indices are stored in different memory locations

Preserves the program semantics

Example 2. DCT-like code

```
do b_r = 0,63
   do b_c = 0,63
      do r = 0,7
         S:A(b_r, b_c, r, *) = ...
      end
   end
end
```

Pipelined with

```
do b_r = 0,63
   do b_c = 0,63
      do c = 0,7
         T: ... = A(b_r, b_c, *, c)
      end
   end
end
```

How to allocate elements of A in local memory, and minimize the size?

$\rightsquigarrow$ Full array: $64 \times 64 \times 8 \times 8 = 2^{18} = 256\text{K}$

$\rightsquigarrow$ **Optimal linear allocation:** 112 elements, $\sigma : \begin{cases} r \bmod 4 \\ 16(b_r + b_c) + 2r + c \bmod 28. \end{cases}$

How a compiler can automatically find a valid allocation?

Main constraints:

- Optimization of the **size of the allocation** (size of the buffer)
- **Simplicity of the addressing function** for implementation aspects

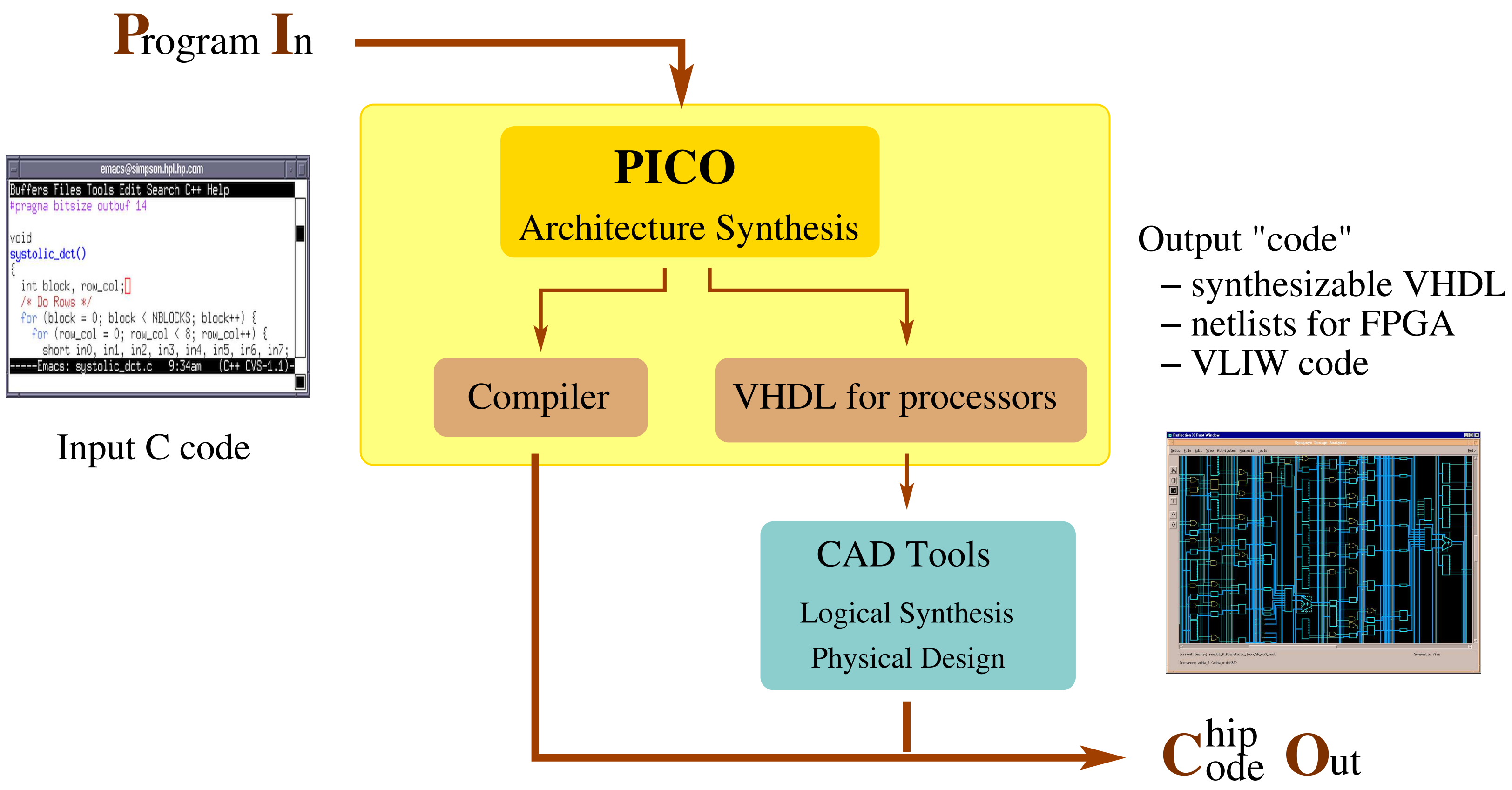
General context:

Compilers, parallelizers (static optimization, loop transformation, . . .)

Application-specific circuit, communicating hardware processes

Automatic synthesis of hardware accelerators

PICO: Program In Chip Out



Similar tools: MMAlpha (INRIA), Atomium (IMEC), Compaan (Leiden)

Outline

Outline

Scheduled program or communicating processes

$+$

Dependence analysis (lifetime)

$+$

Choice of vector indices (e.g., loop indices or array, etc.)

Scheduled program or communicating processes

$+$

Dependence analysis (lifetime)

$+$

Choice of vector indices (e.g., loop indices or array, etc.)

$\downarrow$

Data storage optimization with respect to a representation

Previous approaches

De Greef, Catthoor and De Man (1996-1997)

Lefebvre and Feautrier (1996-1997)

Wilde and Rajopadhye (1996), Quilleré and Rajopadhye (2000)

Strout, Carter, Ferrante and Simon (1998)

Thies, Vivien, Sheldon and Amarasinghe (2001)

All these approaches may be formalized using:

Definition: Two indices $\vec{i}$ and $\vec{j}$ of $\mathbb{Z}^n$ are **conflicting** ($\vec{i} \bowtie \vec{j}$) if they correspond to two values that are simultaneously alive during the execution with schedule θ.

$\mathbf{CS} = \{(\vec{i}, \vec{j})\} | \vec{i} \bowtie \vec{j}\}$: the set of all pairs of conflicting indices.

Ex: $\{(\left[\begin{array}{c}0\\0\end{array}\right], \left[\begin{array}{c}0\\1\end{array}\right]), (\left[\begin{array}{c}0\\1\end{array}\right], \left[\begin{array}{c}0\\2\end{array}\right]), \ldots, (\left[\begin{array}{c}0\\5\end{array}\right], \left[\begin{array}{c}1\\0\end{array}\right]), (\left[\begin{array}{c}1\\5\end{array}\right], \left[\begin{array}{c}2\\0\end{array}\right]), \ldots\}$

Definition: a **linear allocation** of size m is a homomorphism $\sigma : \mathbb{Z}^n \rightarrow \mathcal{M}$, where $\mathcal{M} \subset \mathbb{Z}^p$ is a finite abelian group of m elements.

Valid linear allocation

For conflicting indices $\vec{i}$ and $\vec{j}$, $\vec{i} \neq \vec{j}$ one must have $\sigma(\vec{i}) \neq \sigma(\vec{j})$, i.e. $\sigma(\vec{i} - \vec{j}) \neq 0$

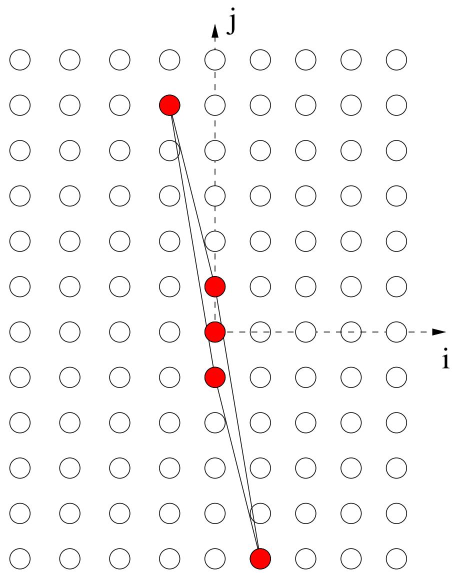
$\mathbf{DS} = \{\vec{i} - \vec{j} \in \mathbb{Z}^n | \vec{i} \bowtie \vec{j}\}$

Definition: σ is **valid** iff $DS \cap \ker \sigma = \{\vec{0}\}$.

Example of Difference Set

(i,j)−(i,j+1)= (0,1)

(0,N−1)−(1,0)=(−1,N−1)



For affine schedules, regular sets of iteration, and affine access functions, CS is represented as all integral points in a union of polytopes.

Depending on the dependence analysis, CS and DS are super-approximated, let $CS \subseteq \mathcal{C}$ and $DS \subseteq \mathcal{D}$.

Let $\mathcal{D}$ be an approximation of the difference set: $DS \subseteq \mathcal{D}$.

$\mathcal{D}$ is the set of **integral points** within a 0-**symmetric polytope** K: $\mathcal{D} = \mathring{K}$ (or a body)

Problem: Minimize the size of linear allocations valid for $\mathcal{D}$ (or K).

Previous heuristics: ex. storage in a 2d buffer

[Successive projections — Lefebvre and Feautrier, loop indices]

[Canonical linearizations — De Greef *et al.*, array indices]

For a given index basis
Choice of **appropriate moduli** such that

$$\sigma(\vec{i}) = \vec{i} \bmod \vec{b} = \begin{bmatrix} i_1 \\ i_2 \end{bmatrix} \bmod \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$$

or one of the $2^n n!$ canonical linearization

$$\sigma(\vec{i}) = \pm N i_1 \pm i_2 \bmod b \quad \text{or} \quad \sigma(\vec{i}) = \pm i_1 \pm N i_2 \bmod b$$

is a valid allocation.

Ex. σ **must be nonzero** on $\mathcal{D} = \{(0,1), (1, 1-N), \ldots\}$

Largest component along e_1:

$$\begin{bmatrix} 1 & 0 \\ \cdot & \cdot \end{bmatrix} \begin{bmatrix} 1 \\ 1-N \end{bmatrix} = \begin{bmatrix} 1 \\ \cdot \end{bmatrix}$$

Largest component in the orthogonal:

$$\begin{bmatrix} \cdot & \cdot \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

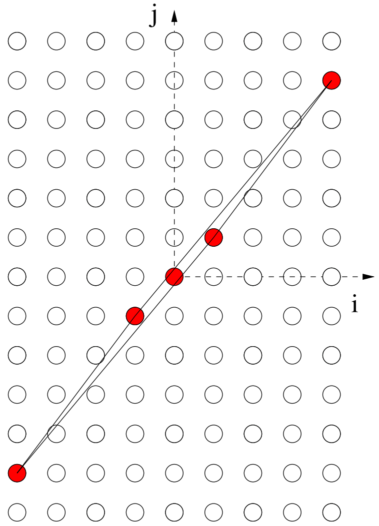
$$\Rightarrow \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} i_1 \\ i_2 \end{bmatrix} \bmod \begin{bmatrix} 2 \\ 2 \end{bmatrix}$$

Size $=$ **4**

or best canonical linearization

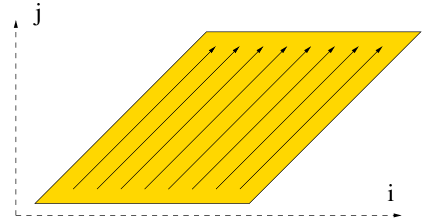
$\max_{\mathcal{D}}\{Ni + j\} = 1 \Rightarrow Ni + j \bmod 2,$ Size $=$ **2**

Limitations



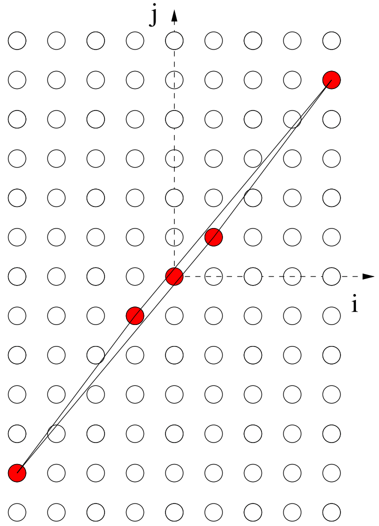
Optimal size: 2
(unchanged)

New schedule: θ(i,j)=(i−j,i)



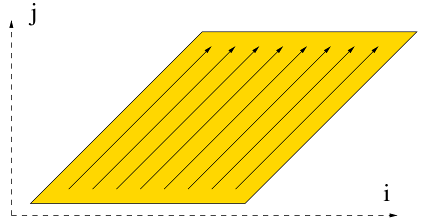
$$\mathcal{D} = \{(1,1),(N-1,N),\ldots\}$$

Limitations



Optimal size: 2
(unchanged)

New schedule: θ(i,j)=(i–j,i)



$$\mathcal{D} = \{(1,1),(N-1,N),\ldots\}$$

$$\begin{bmatrix} 1 & 0 \\ . & . \end{bmatrix} \begin{bmatrix} N-1 \\ N \end{bmatrix} = \begin{bmatrix} N-1 \\ . \end{bmatrix} \quad \Rightarrow \quad \text{modulo } N \quad \Rightarrow \quad \text{Size} = \mathbf{N}$$

or

$$\max_{\mathcal{D}}\{|\pm Ni \pm j|\} = \max_{\mathcal{D}}\{|\pm i \pm Nj|\} = N(N-1) \quad \Rightarrow \quad \text{Size} = \mathbf{O(N^2)}$$

Our contribution

- ▷ **Geometrical framework** for formalizing and studying heuristics
- ▷ **Lower and upper bounds on performance** with respect to $\mathcal{D}$ and K
- ▷ **Guaranteed heuristics**, i.e., whose size cannot be "arbitrarily bad"

Outline

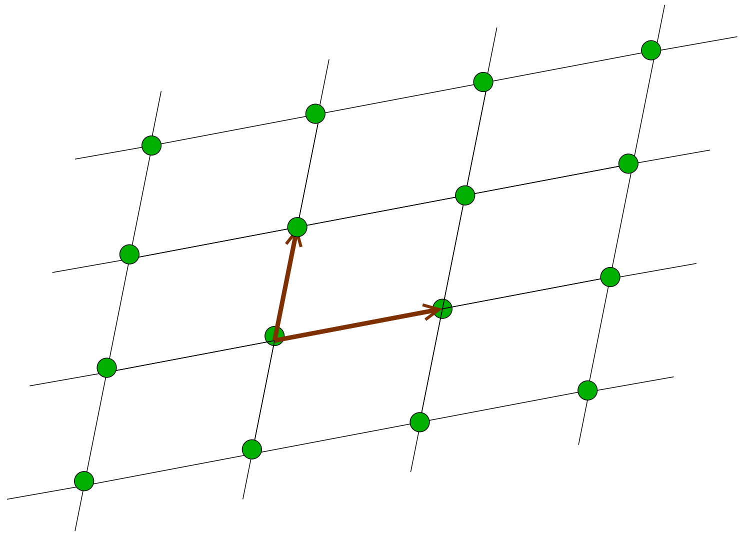
Geometrical interpretation

[Early work on skewing schemes: Budnik and Kuck 1971, Shapiro 78, Wijshoff and Van Leeuwen 1985]

Validity: $K \cap \ker \sigma = \{\vec{0}\}$

Kernel of σ: $\vec{i}, \vec{j} \in \ker \sigma \subset \mathbb{Z}^n \Rightarrow \quad u\vec{i} + v\vec{j} \in \ker \sigma,\ u, v \in \mathbb{Z}.$

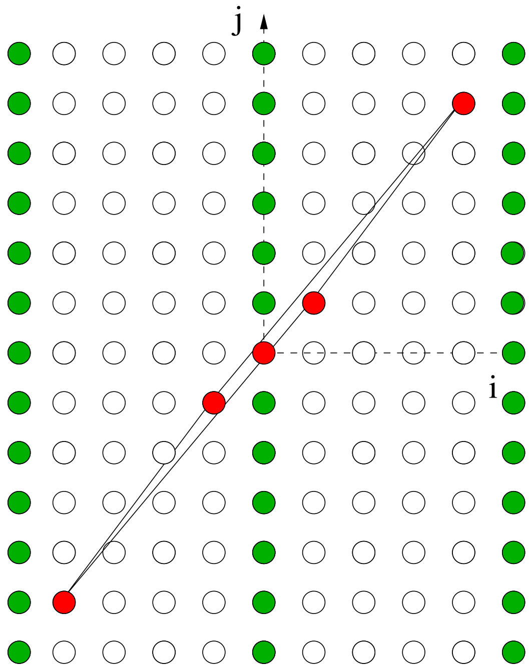
The kernel of a linear allocation is an integral lattice



Validity $\equiv$ strictly admissible lattice

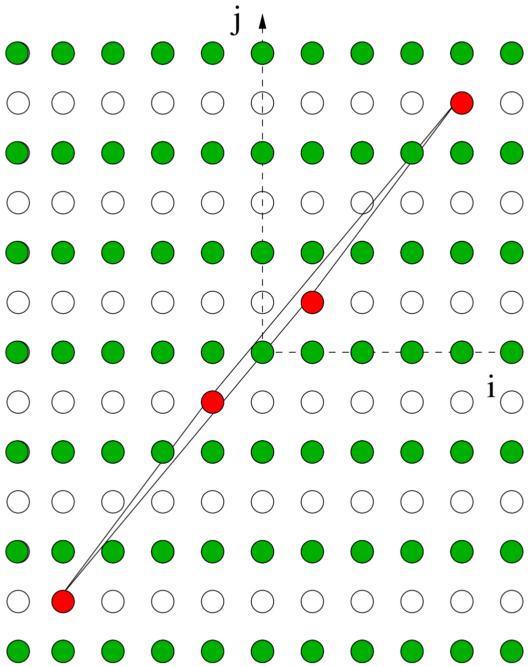
Definition: The lattice $\Lambda = \ker \sigma$ is **strictly admissible** for the polytope K iff

$$K \cap \Lambda = \{\vec{0}\}$$



Size N allocation

“Good” allocation ≡ “accurate” strictly admissible lattice



Optimal size: 2

Up to equivalence (same kernel),

$$\sigma : \vec{i} \mapsto U \cdot \vec{i} \bmod \vec{s} = \begin{cases} u_{11}i_1 + \ldots + u_{1n}i_n \bmod s_1 \\ \ldots \\ u_{n1}i_1 \ldots + u_{nn}i_n \bmod s_n \end{cases}$$

with U unimodular and diag($\vec{s}$) in Smith normal form.

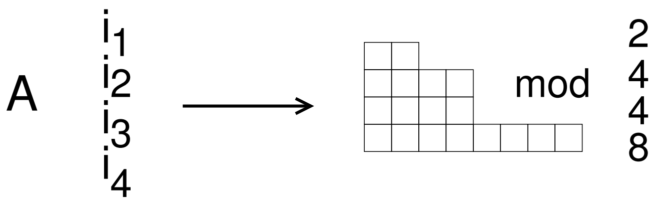
Up to equivalence (same kernel),

$$\sigma : \vec{i} \mapsto U \cdot \vec{i} \bmod \vec{s} = \begin{cases} u_{11}i_1 + \ldots + u_{1n}i_n \bmod s_1 \\ \ldots \\ u_{n1}i_1 \ldots + u_{nn}i_n \bmod s_n \end{cases}$$

with U unimodular and diag($\vec{s}$) in Smith normal form.

Storage

Size $= s_1 s_2 \ldots s_n$



Size = 256 (dim 4)

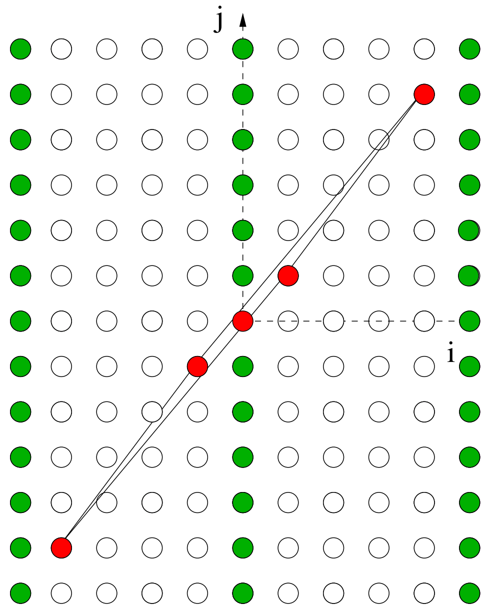
Underlying lattice Λ (the kernel)

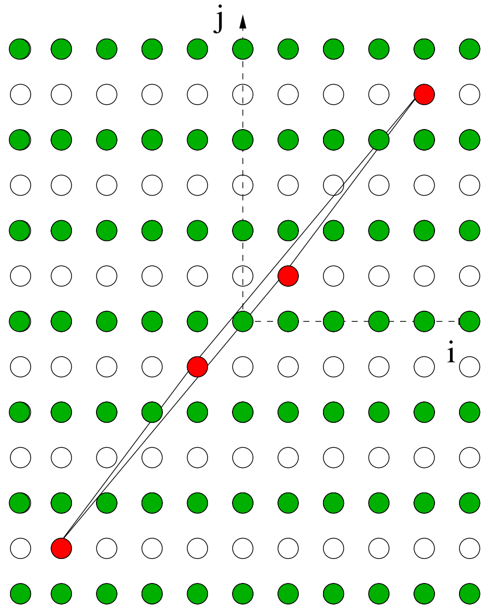
det $\Lambda = s_1 s_2 \ldots s_n$

$$\Lambda : \ U^{-1} \begin{bmatrix} s_1 & & \\ & \ldots & \\ & & s_n \end{bmatrix}$$

$$\Lambda : \begin{bmatrix} N & 0 \\ 0 & 1 \end{bmatrix}, \ \det \Lambda = \mathbf{N}$$

$$\Lambda : \begin{bmatrix} 1-N & 2 \\ -N & 2 \end{bmatrix}, \ \det \Lambda = \mathbf{2}$$





Surface = N

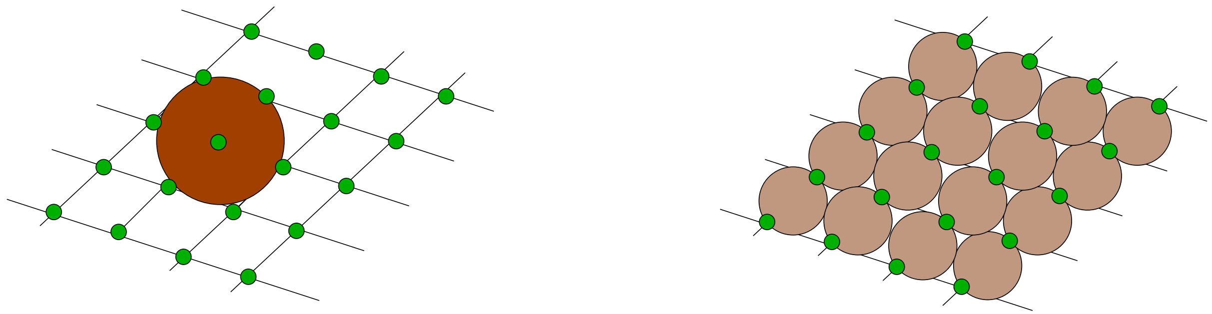
Surface = 2

K a 0-symmetric polytope (or a body)

Problem: Find a lattice, **integral** and **strictly admissible** for K, of **small determinant**

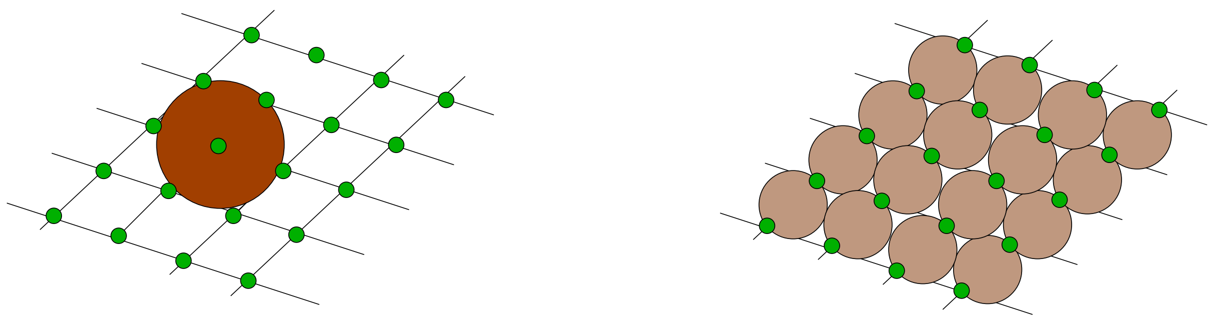
Nota: then, one constructs a valid allocation whose kernel is Λ (always possible).

Admissible lattice and lattice packing



Admissible lattice for K ⟷ Lattice packing for K/2

Admissible lattice and lattice packing



Admissible lattice for K $\longleftrightarrow$ Lattice packing for K/2

Density of a lattice packing of K:

$$\delta(K,\Lambda) = \frac{\mathrm{Vol}(K)}{\det \Lambda}$$

Hard question: densest lattice packing? [Rogers 64, Gruber and Lekkerkerker 87]

The **critical determinant** of K:

$\Delta(K) = \inf_{\Lambda}\{ \det \Lambda \mid \Lambda$ is admissible for K $\}$

[Minkowski 1rst Theorem, Minkowski-Hlawka]

$$\frac{\mathsf{Vol}(K)}{2^n} \leq \Delta(K) \leq \mathsf{Vol}(K)$$

The **critical determinant** of K:

$\Delta(K) = \inf_\Lambda \{ \det \Lambda \mid \Lambda \text{ is admissible for } K \}$

[Minkowski 1rst Theorem 1893, Minkowski-Hlawka]

$$\frac{\mathrm{Vol}(K)}{2^n} \leq \Delta(K) \leq \mathrm{Vol}(K)$$

Best memory allocation (linear):

$\Delta_{\mathbb{Z}}(K) = \inf_{\Lambda \text{ integral}} \{ \det \Lambda \mid \Lambda \text{ is strictly admissible for } K \}$

$$\frac{\mathbf{Vol}(K)}{2^n} \leq \Delta_{\mathbb{Z}}(K) \leq \mathbf{?}$$

Outline

Scheme I

Input: K
Output: an integral lattice Λ, strictly admissible for K

1. Start from an integral lattice with basis $(\vec{c}_1, \ldots, \vec{c}_n)$

2.

3. Compute appropriate integer **scaling factors** ρ_i, $1 \leq i \leq n$

Return the lattice with basis $(\rho_1 \vec{c}_1, \ldots, \rho_n \vec{c}_n)$

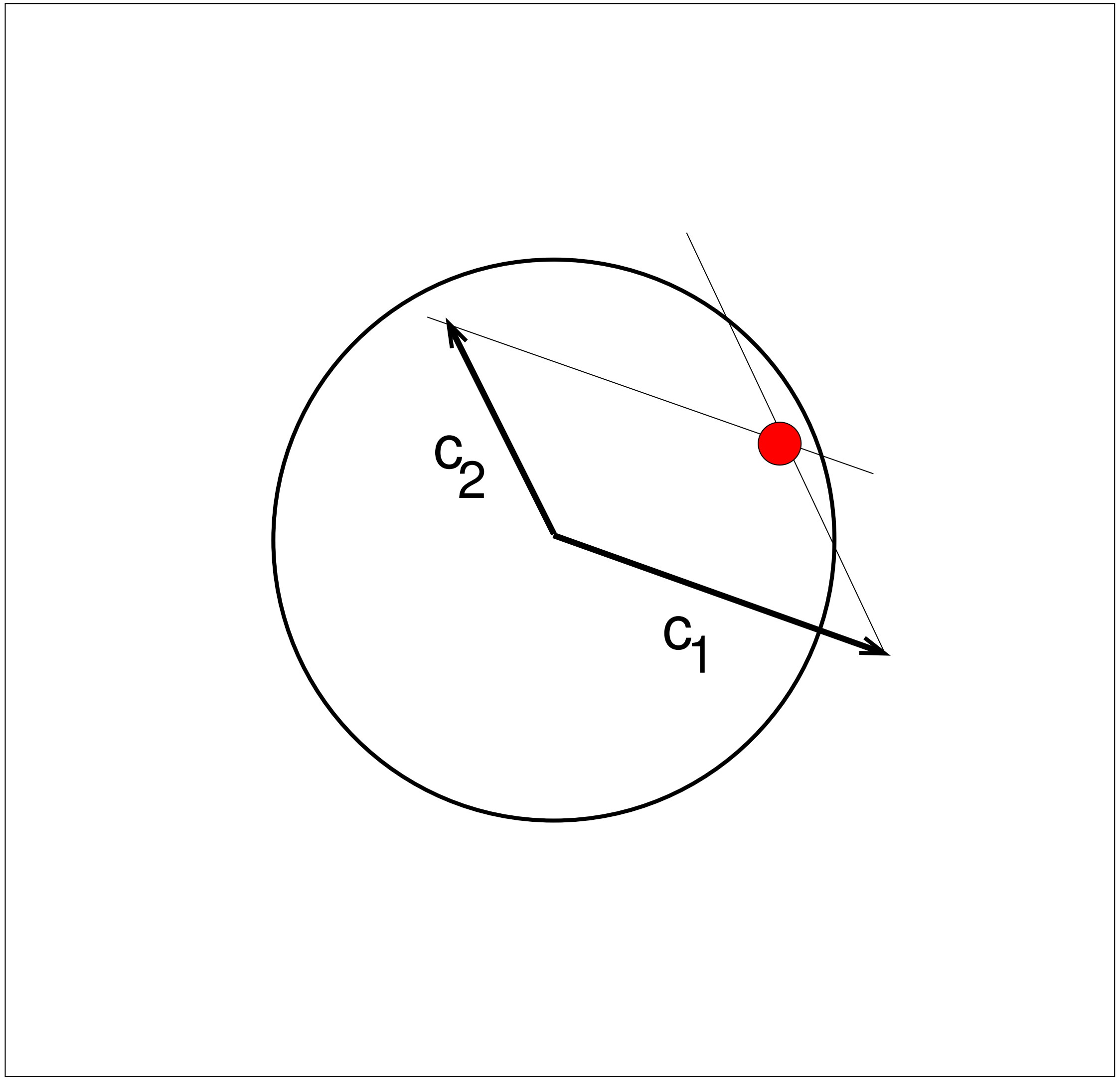
Scheme I

Input: K
Output: an integral lattice Λ, strictly admissible for K, $\det(\Lambda) \leq c_n \mathbf{Vol}(K)$

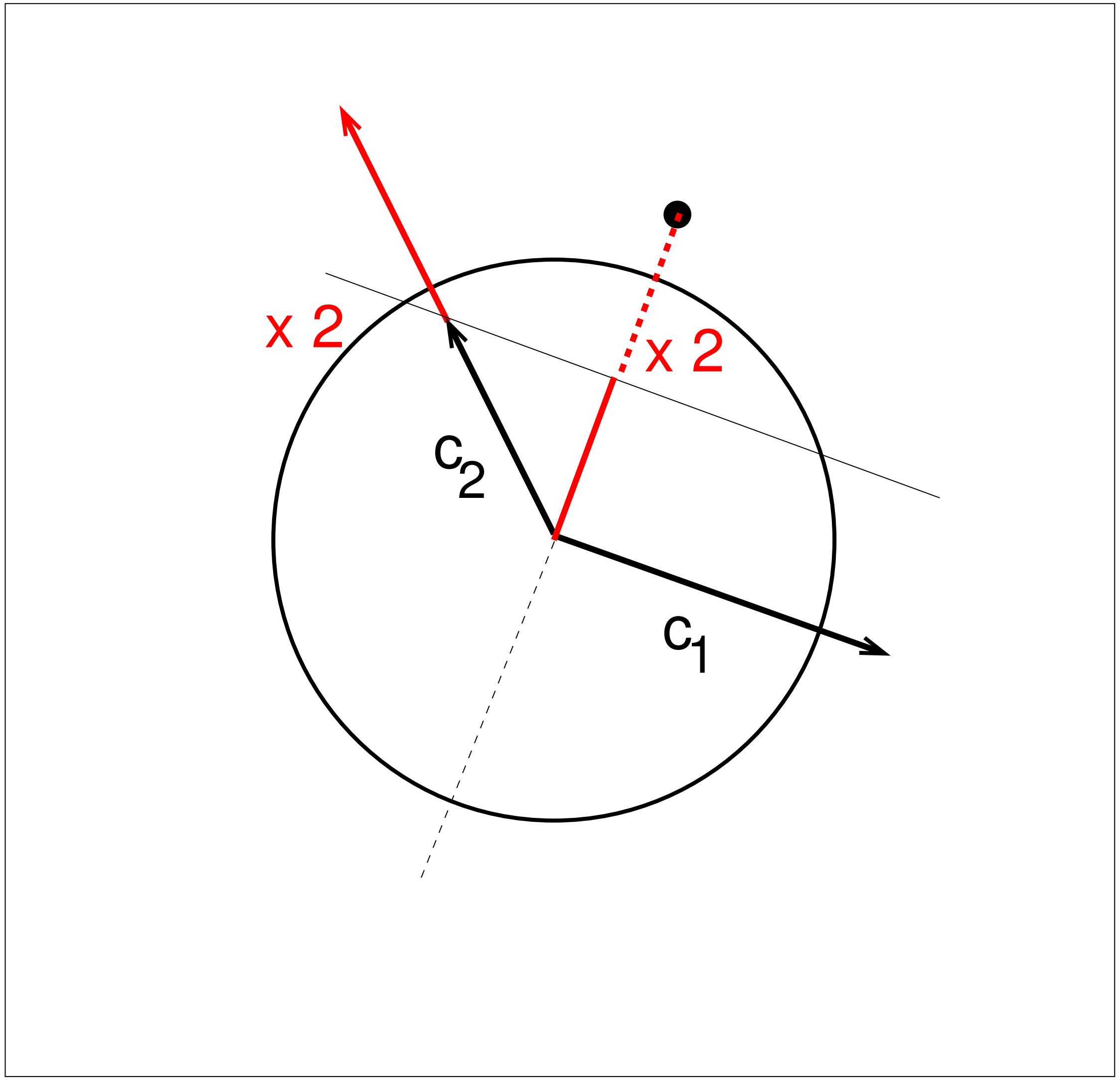
1. Start from an integral lattice with basis $(\vec{c}_1, \ldots, \vec{c}_n)$
2. **"Improve" the basis**
3. Compute appropriate integer **scaling factors** ρ_i, $1 \leq i \leq n$

Return the lattice with basis $(\rho_1 \vec{c}_1, \ldots, \rho_n \vec{c}_n)$

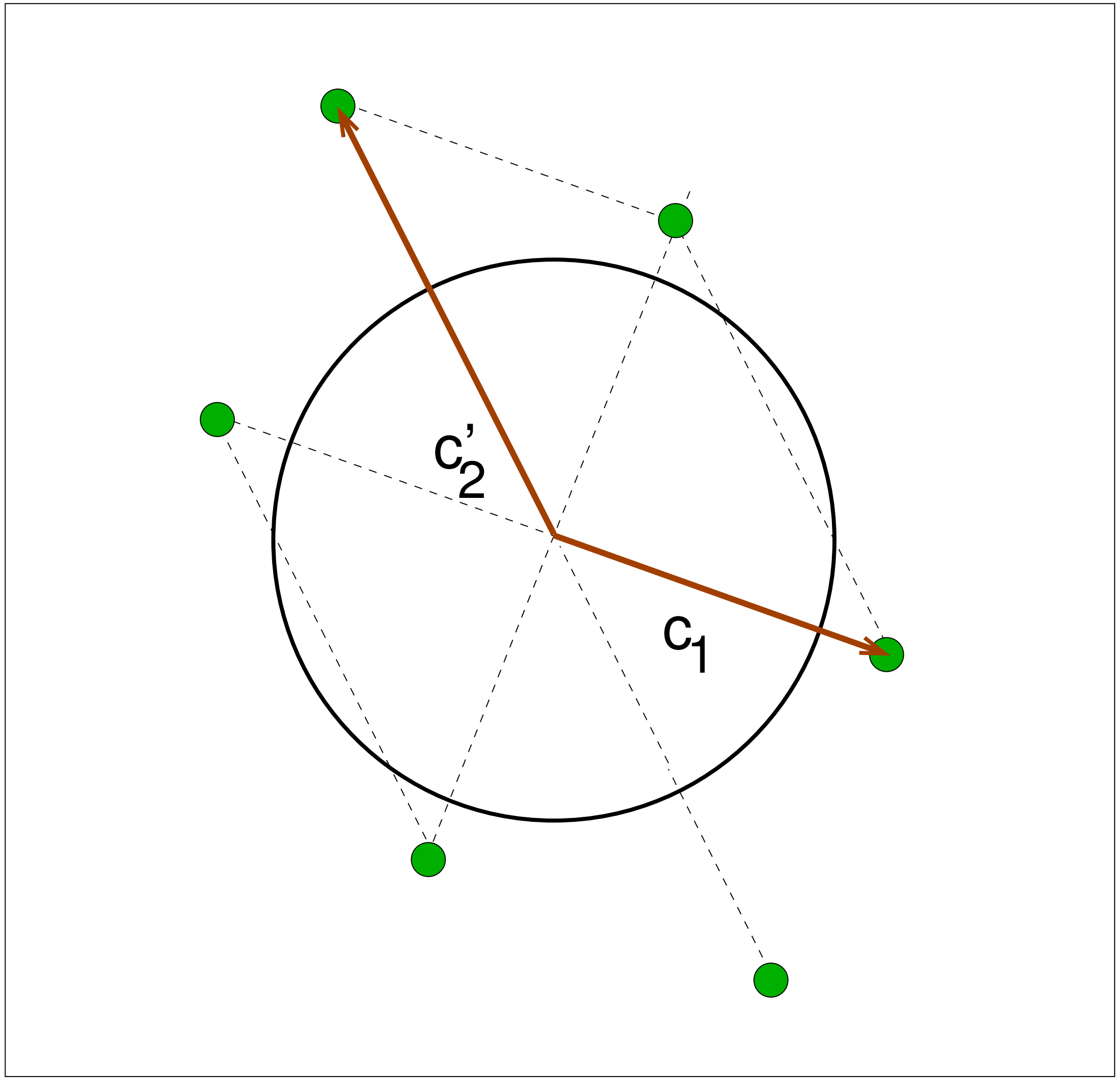
Arbitrary basis for the ball



Arbitrary working basis for the ball

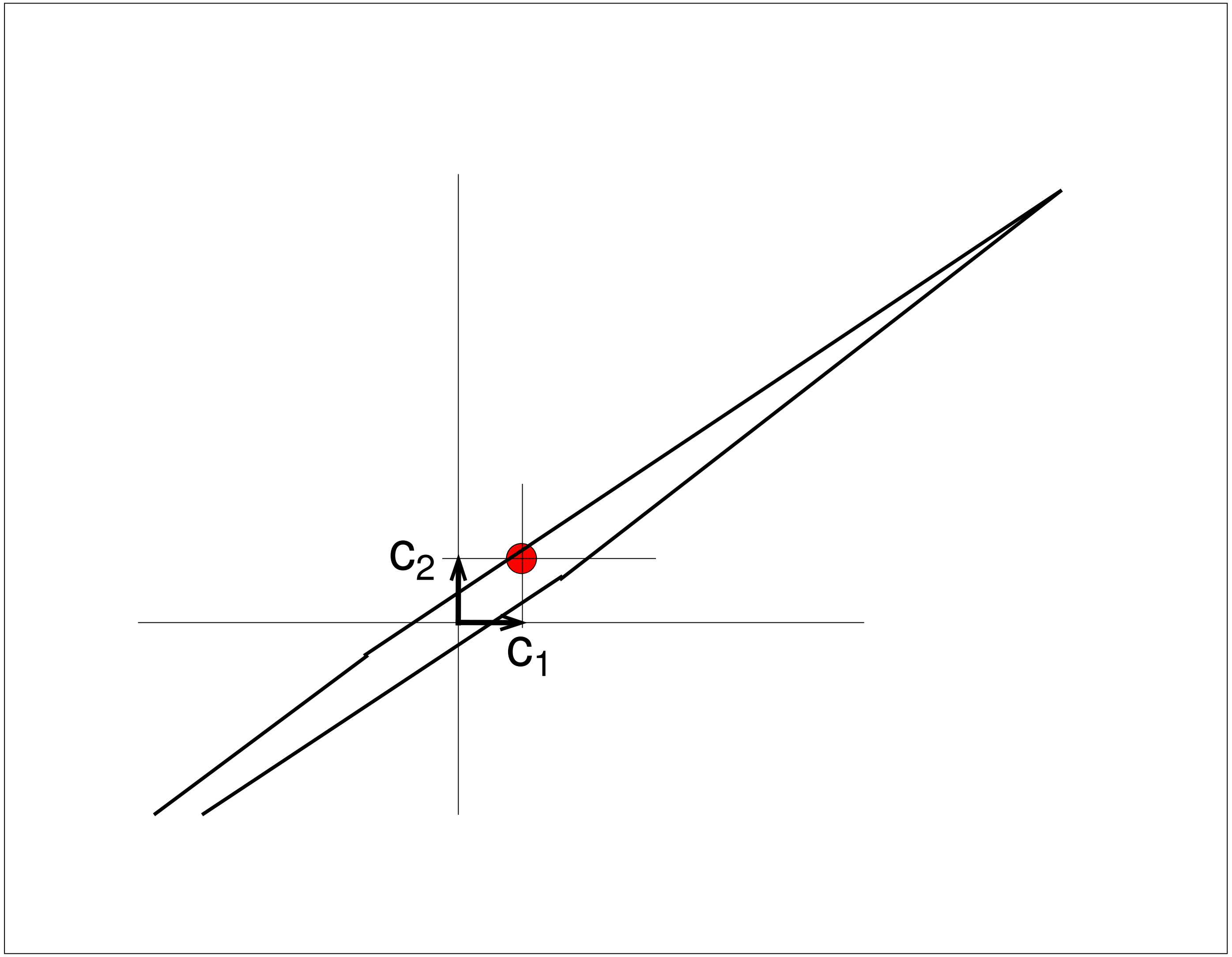


Arbitrary working basis for the ball

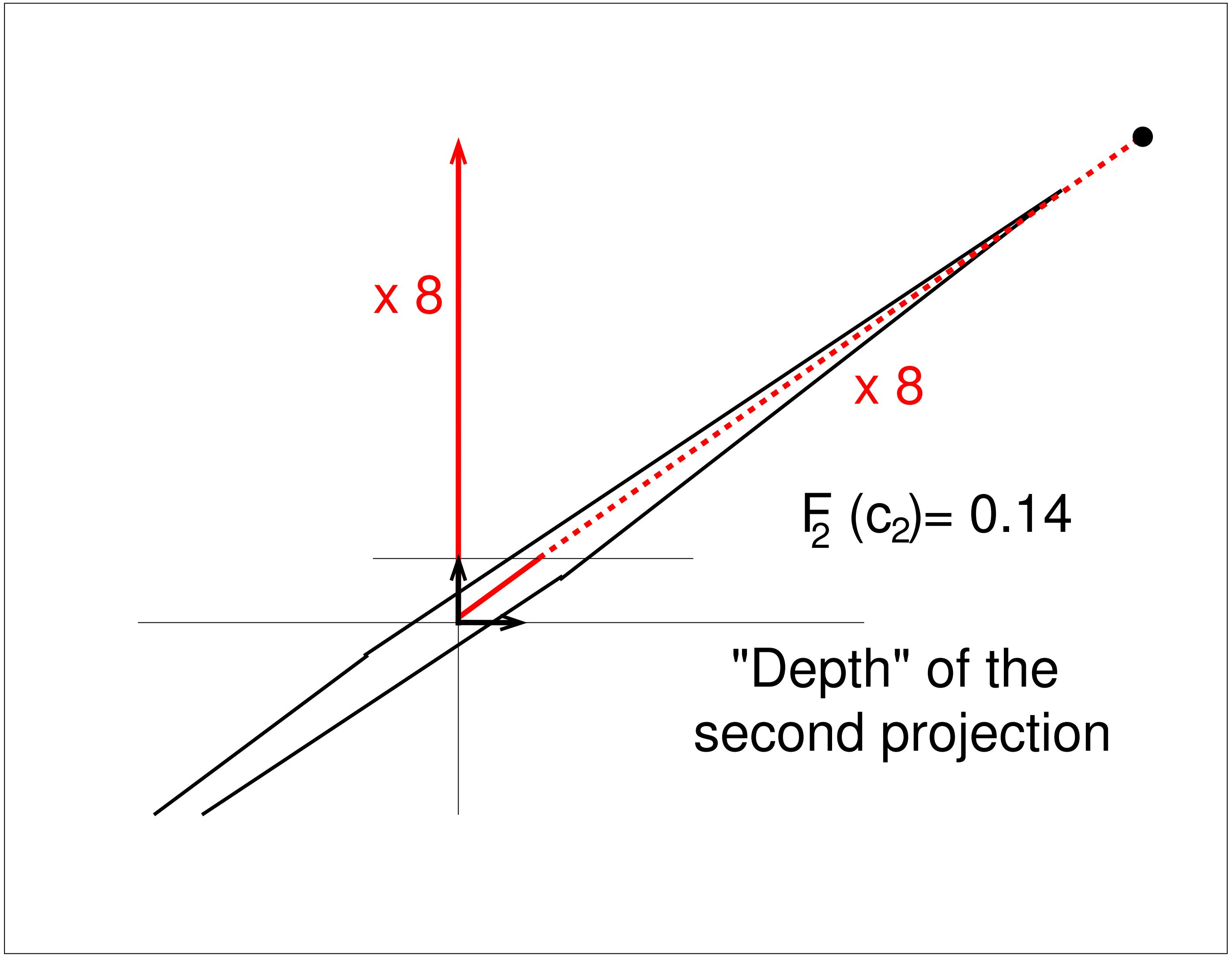


Arbitrary working basis for the ball

Arbitrary basis for a polytope



Arbitrary working basis for a polytope



Arbitrary working basis for a polytope

Definition: ith "depth" [Lovász and Scarf 1992]

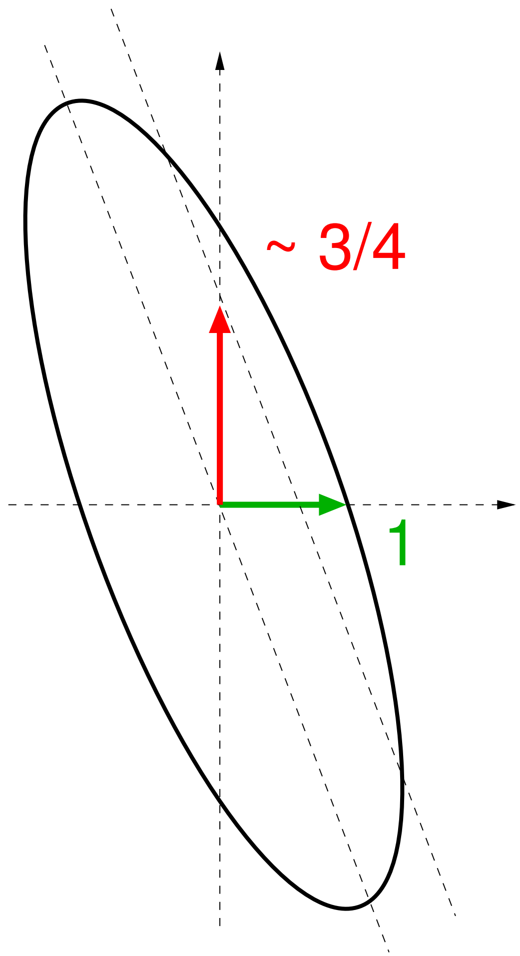
- $F(\vec{c}) = \inf\{\ \rho > 0 \mid \vec{c} \in \rho K\ \}$

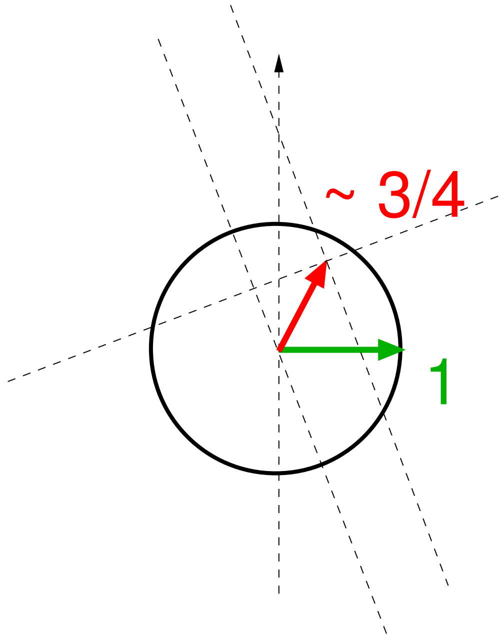
- $F_i(\vec{c}_i) = \inf\{\ F(\vec{x}) \mid \vec{x} \in \vec{c}_i + \mathsf{Vect}(\vec{c}_1, \ldots, \vec{c}_{i-1})\ \}$

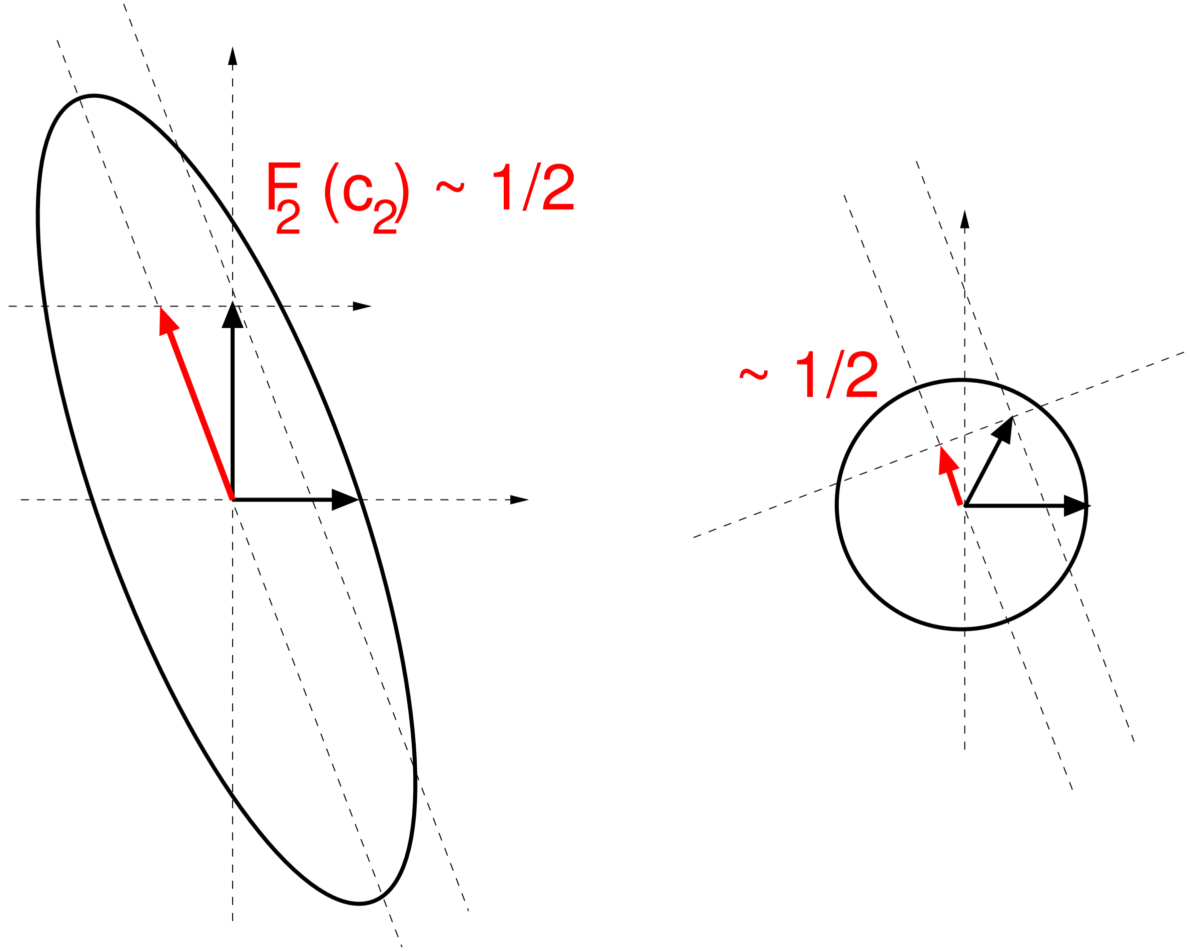
or, working in the dual K^* of K,

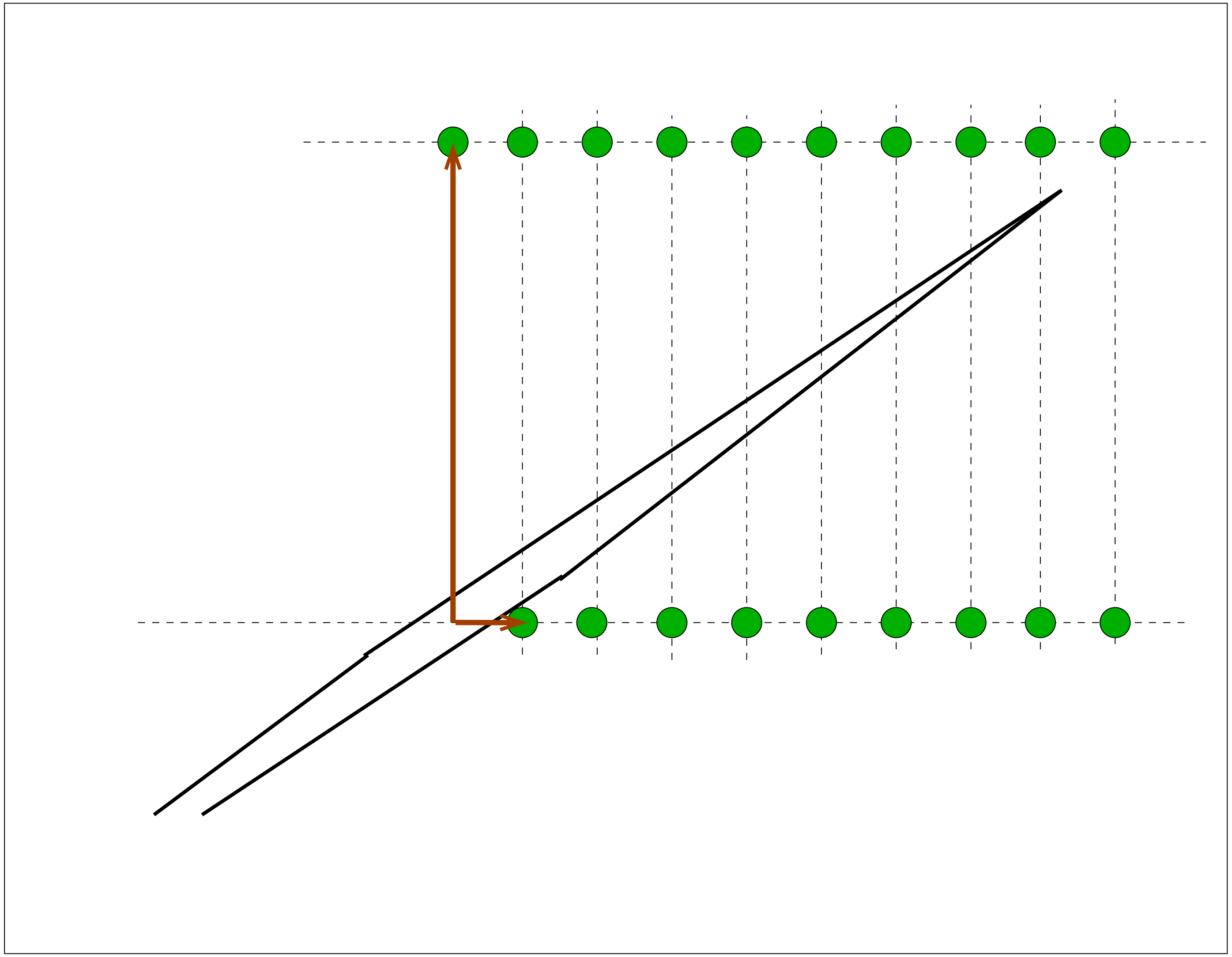
Definition: ith "width"

- $F_i^*(\vec{c}_i) = \sup\{\ \vec{c}_i \cdot \vec{y} \mid \vec{y} \in K, \vec{y} \cdot \vec{c}_1 = \vec{y} \cdot \vec{c}_2 = \ldots = \vec{y} \cdot \vec{c}_{i-1} = 0\}$


~ 3/4
1


~ 3/4
1


F2 (c2) ~ 1/2
~ 1/2



Arbitrary working basis for a polytope

What's wrong with the working basis?

The determinant of the output basis is related to

$$\prod_{i=1}^{n} \rho_i \approx \prod_{i=1}^{n} \frac{1}{F_i(\vec{c}_i)},$$

hence, for upper bounding the determinant,

$$\prod_{i=1}^{n} F_i(\vec{c}_i) \geq ?$$

The determinant of the output basis is related to

$$\prod_{i=1}^{n} \rho_i \approx \prod_{i=1}^{n} \frac{1}{F_i(\vec{c}_i)},$$

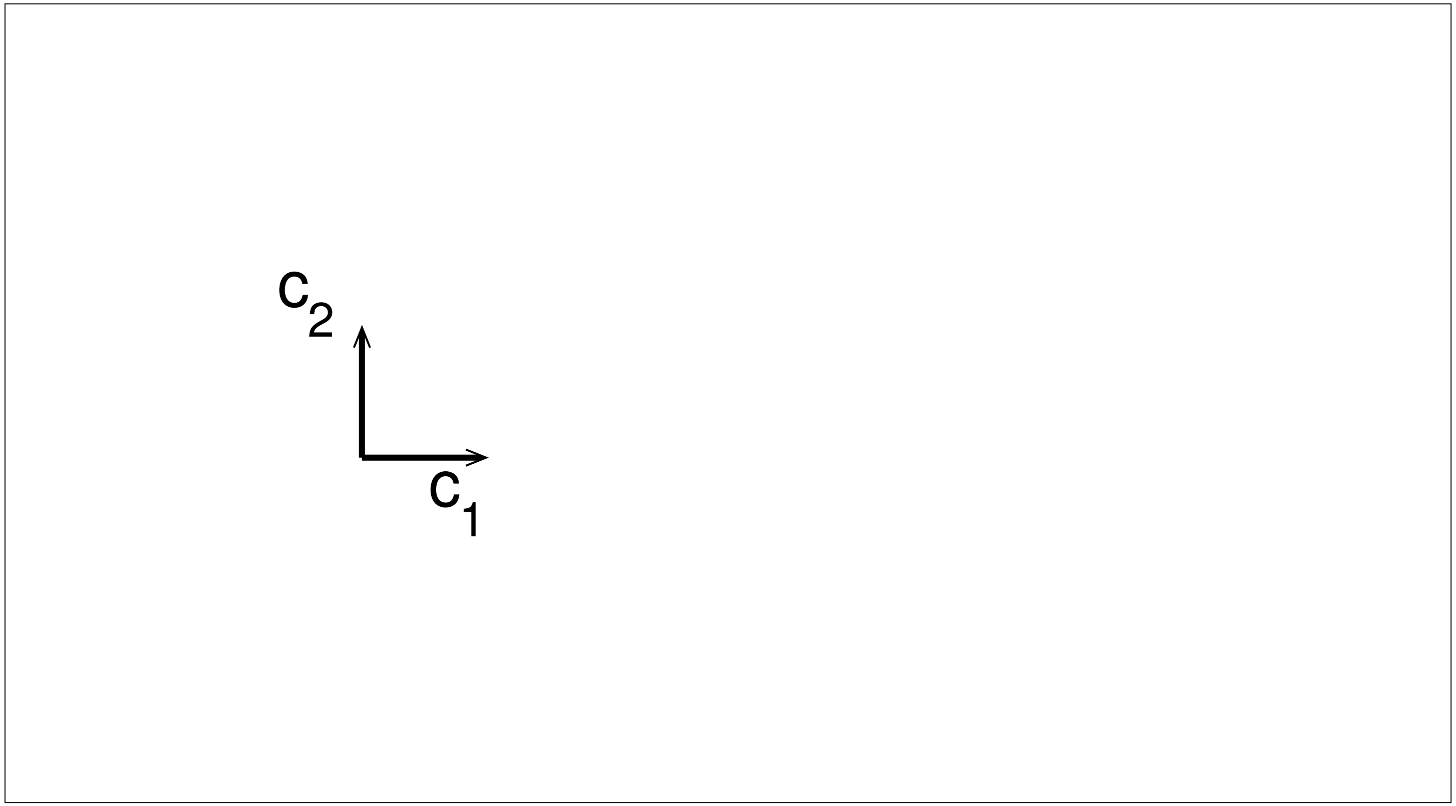
hence, for upper bounding the determinant,

$$\prod_{i=1}^{n} F_i(\vec{c}_i) \geq ?$$

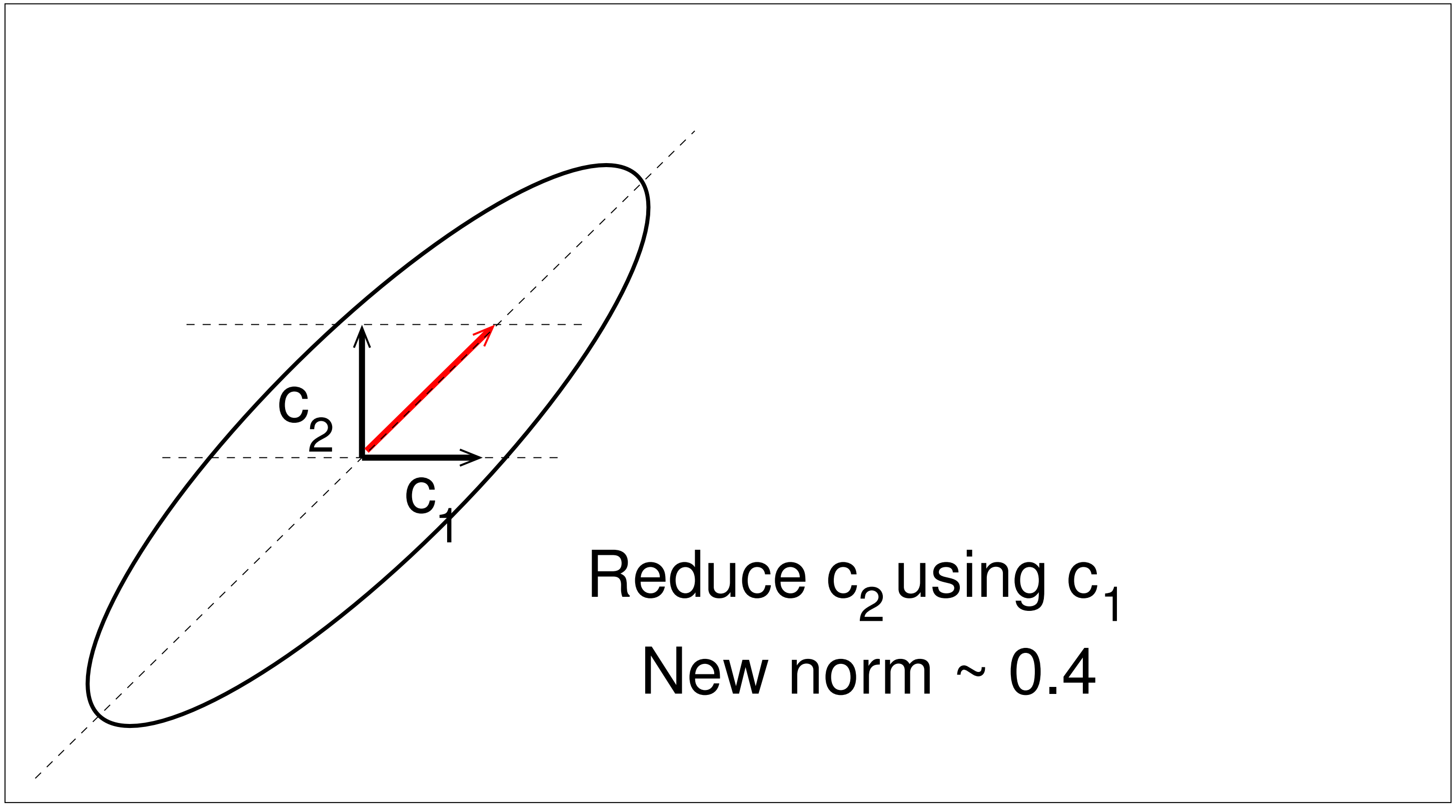
$\Rightarrow$ use a **generalized reduced basis** [Lovász and Scarf 92] with

$$F_i(\vec{c}_i) \geq \lambda_i(K)(\frac{1}{2} - \epsilon)^{i-1}$$

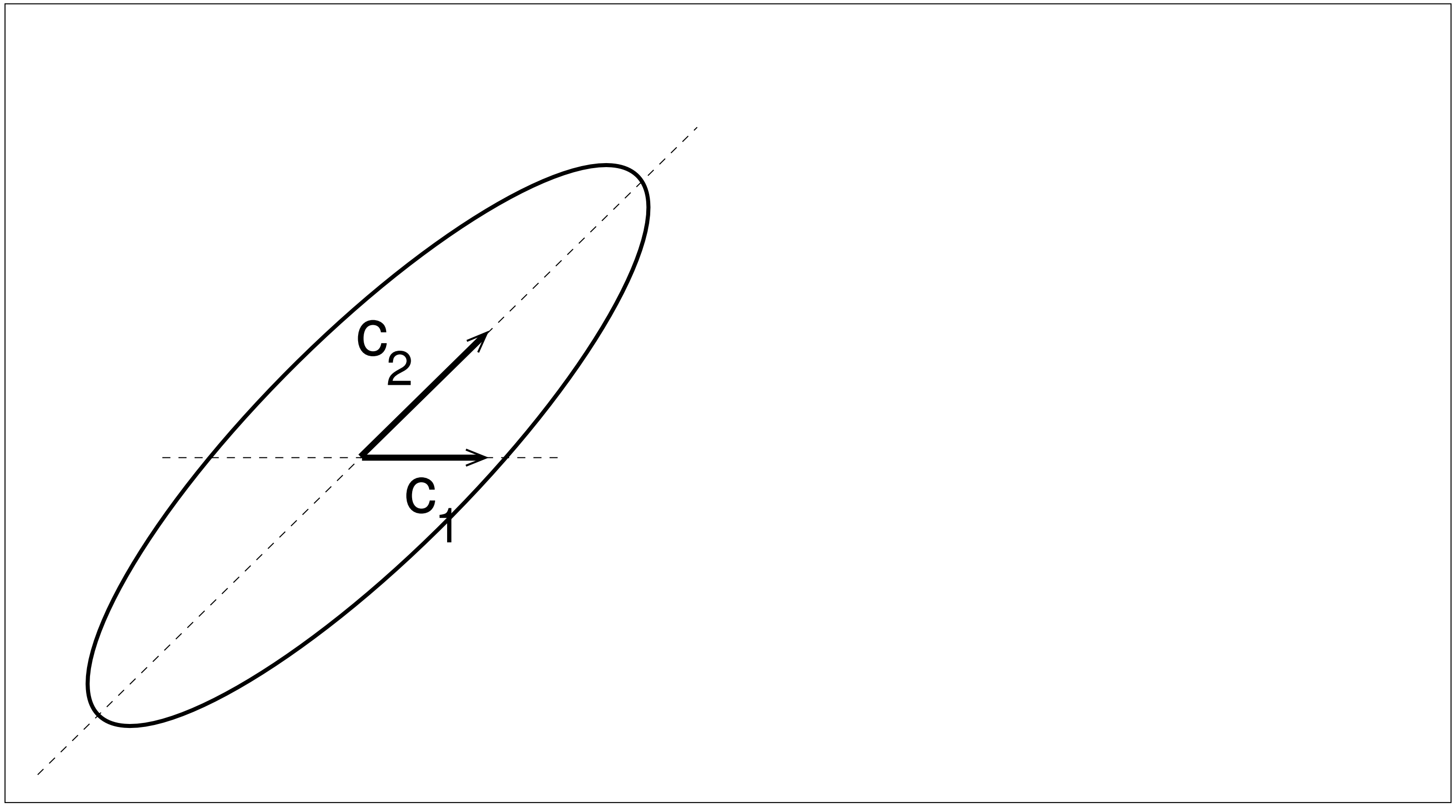
and for the successive minima $\lambda_i(K)$, use the Second Theorem of Minkowski.



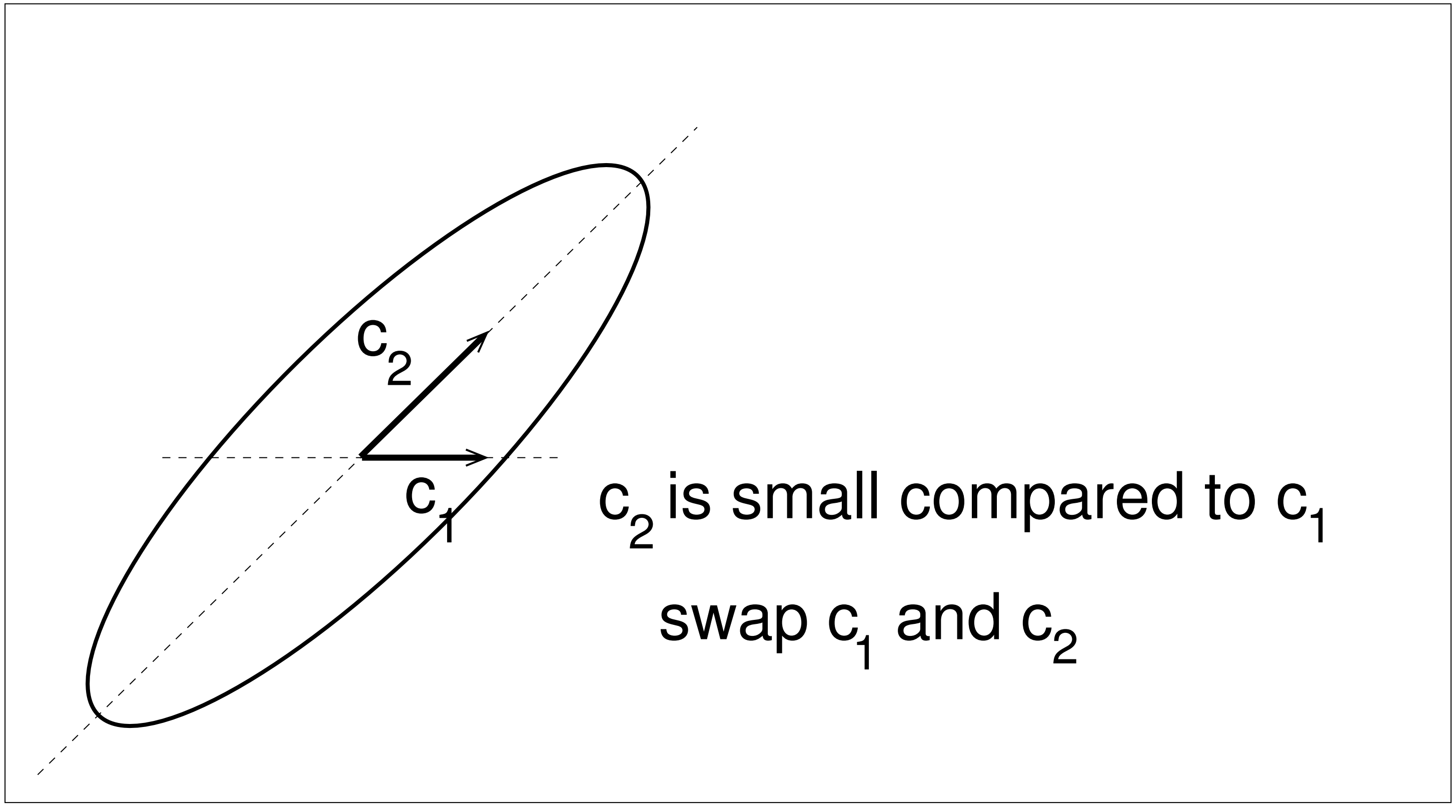
Generalized lattice basis reduction



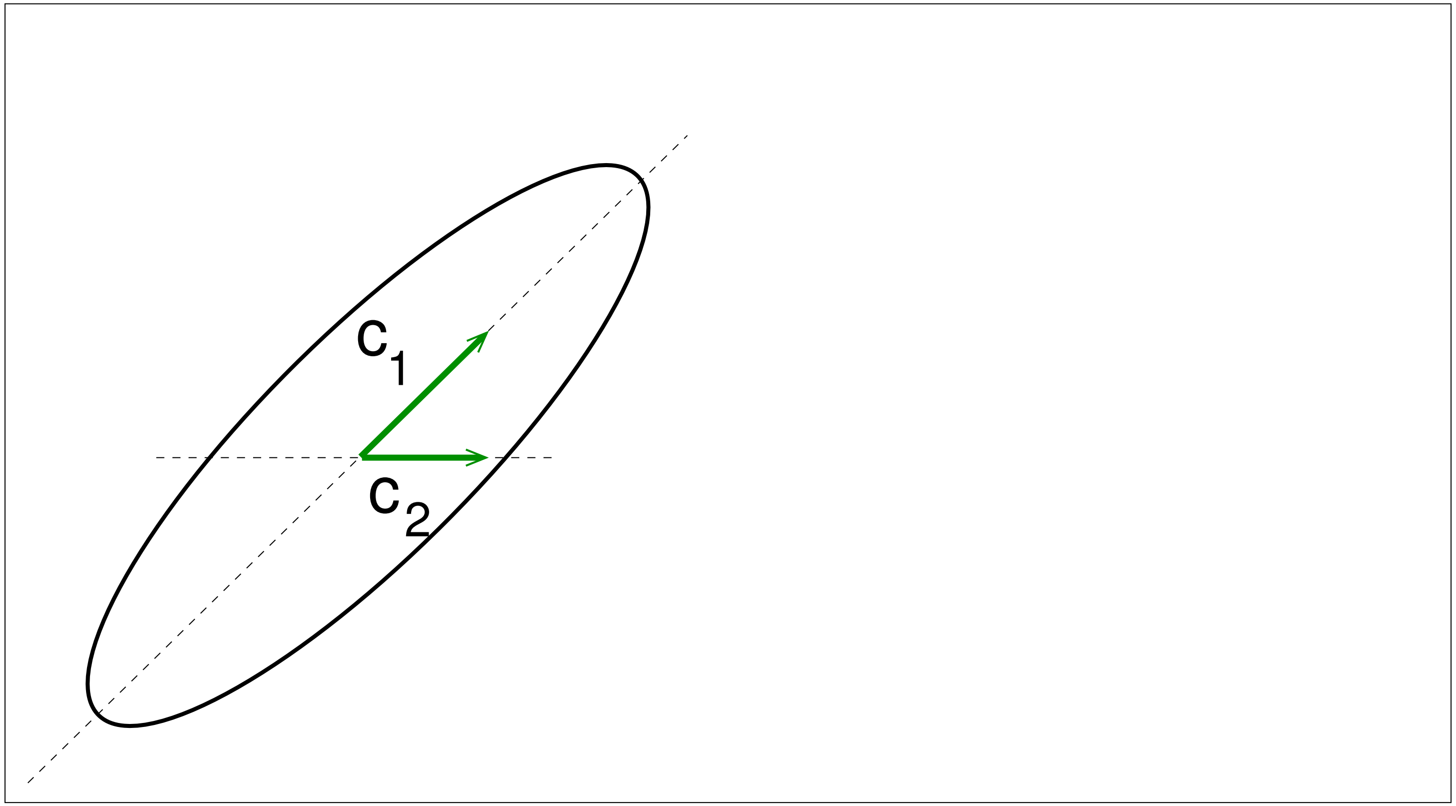
Generalized lattice basis reduction



Generalized lattice basis reduction



Generalized lattice basis reduction



Generalized lattice basis reduction

Application to memory allocations

1. **Better understanding of previous heuristics**

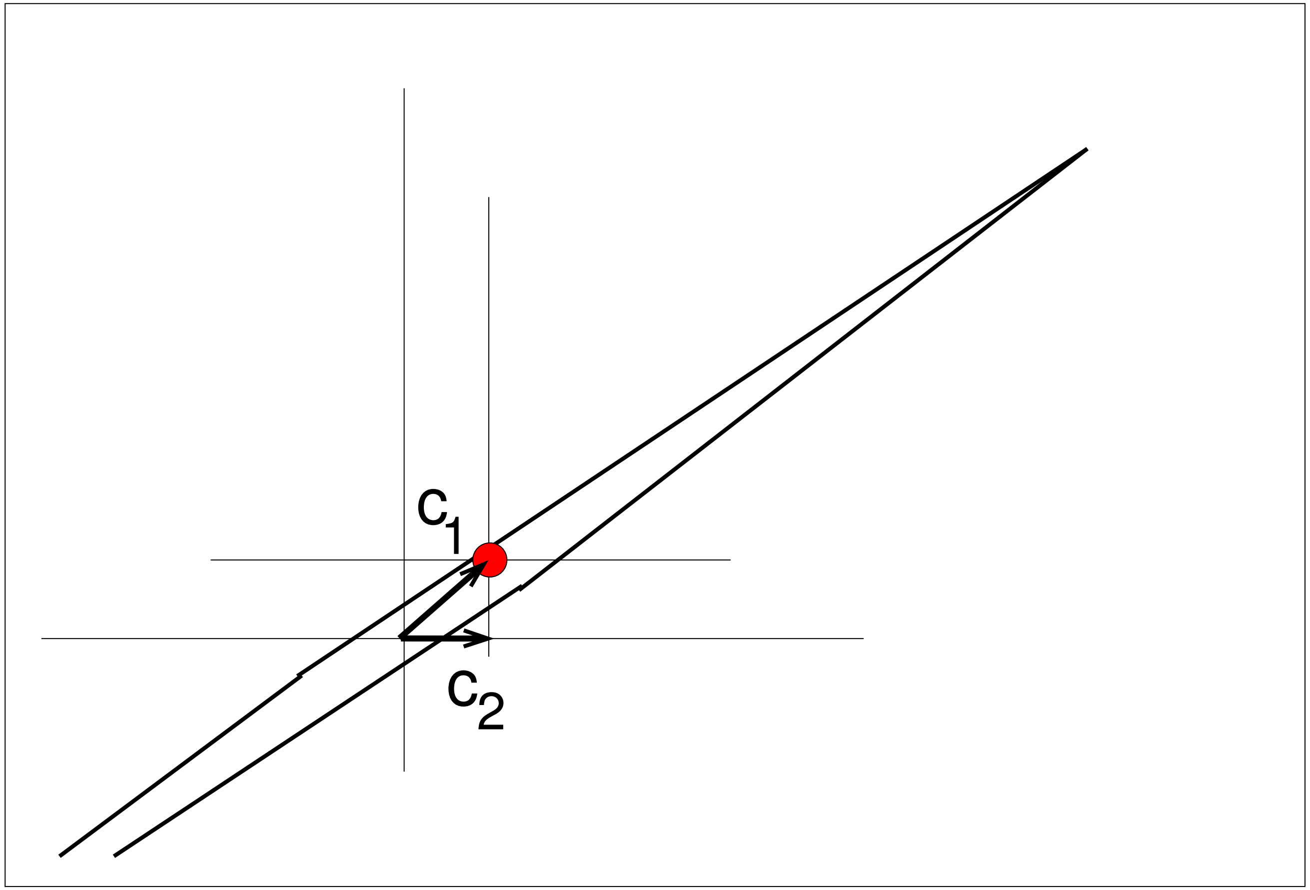
Based on “fixed” bases (loops, arrays, schedule, . . .)

$\Rightarrow$ may fail if the basis is not adequate with respect to DS

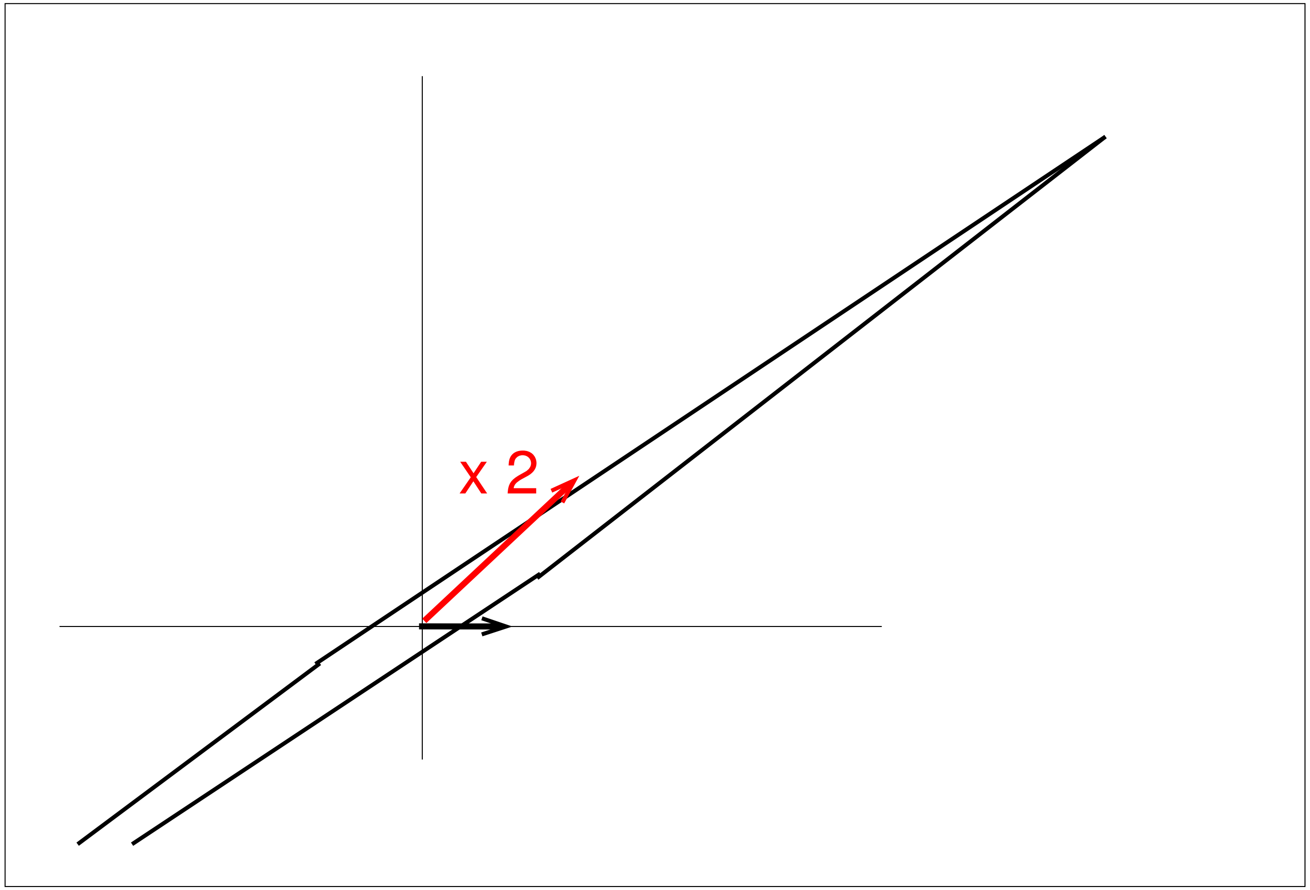
2. **Upper bound for the strictly admissible determinant $\Delta_{\mathbb{Z}}$**

3. **Provides heuristics with guaranteed size**

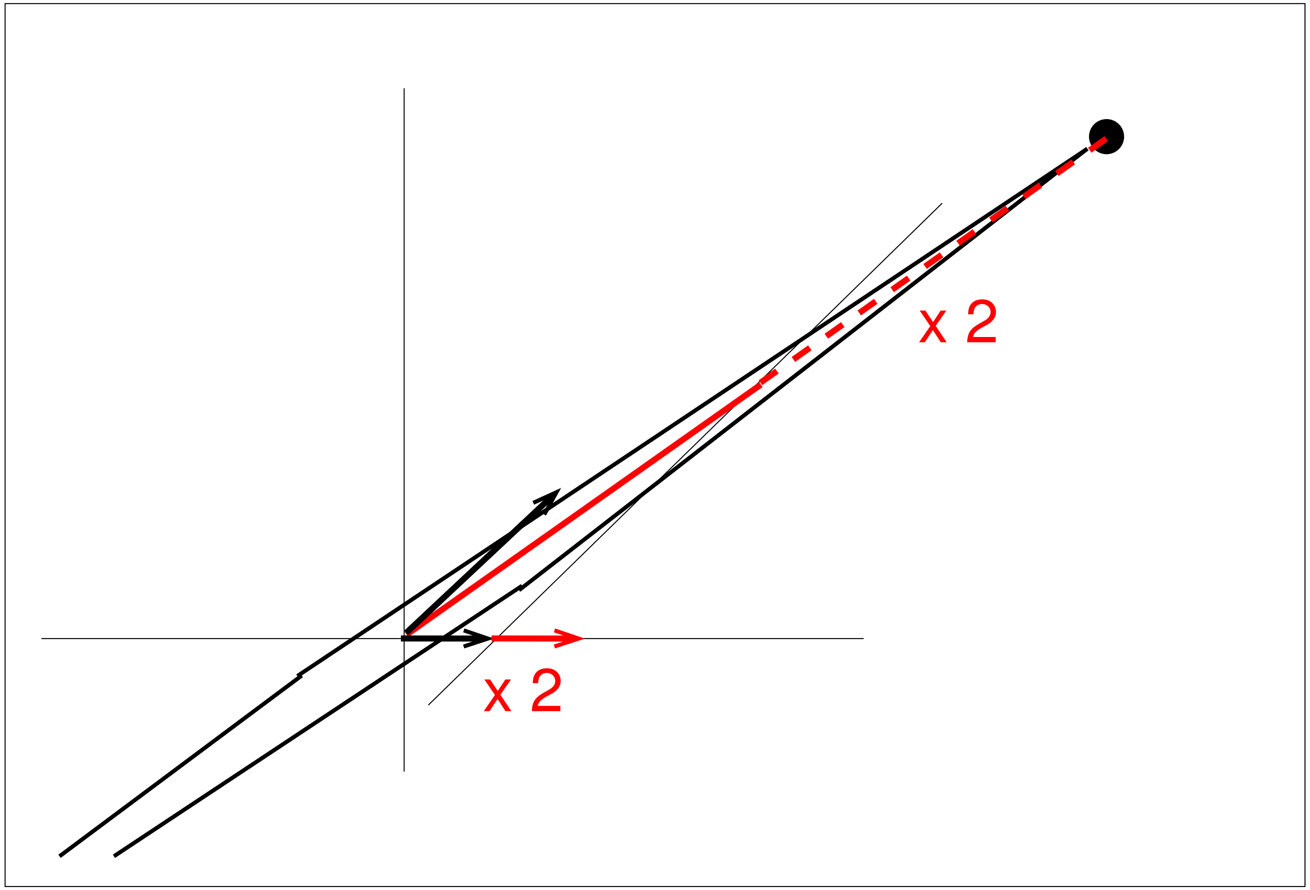
Improved basis for a polytope



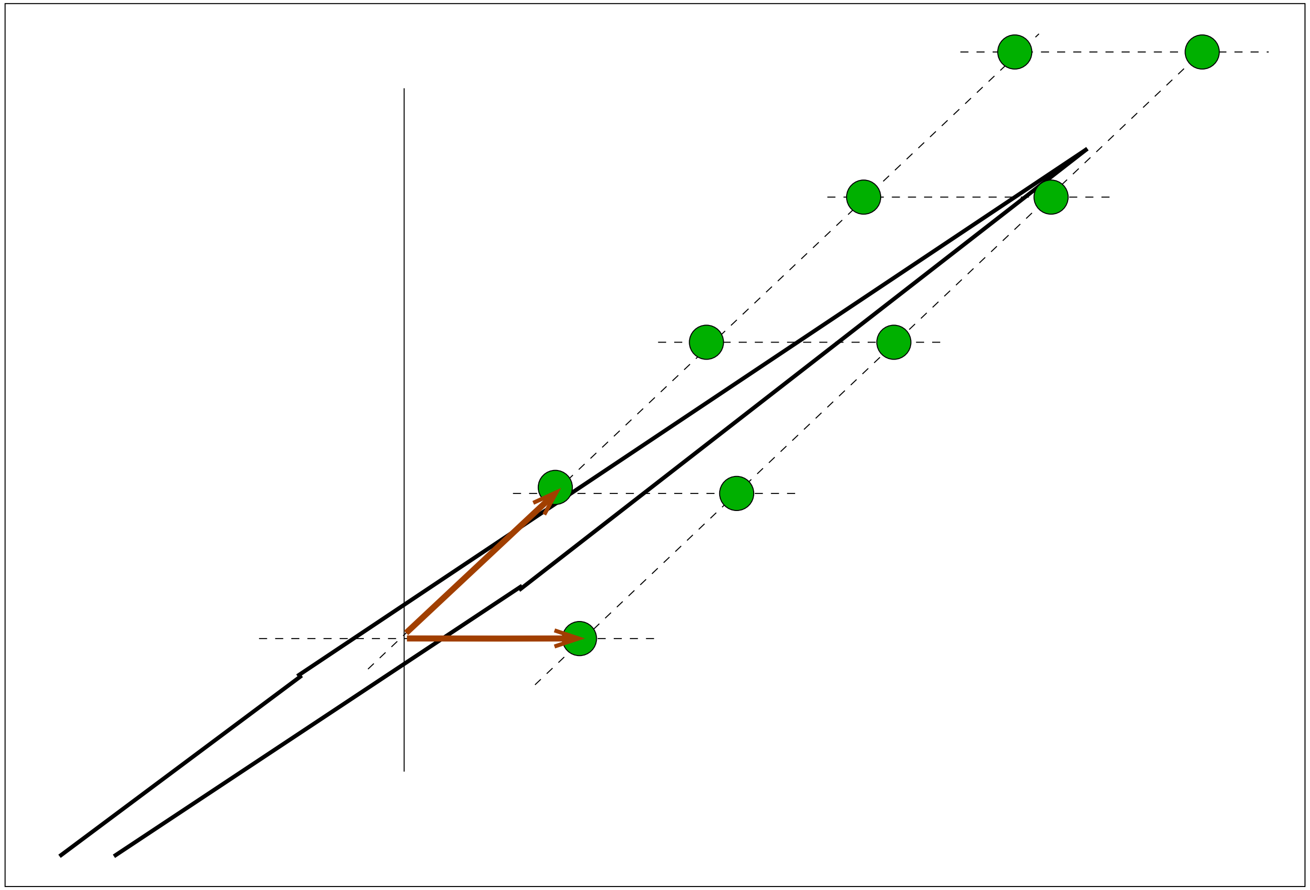
Improved basis for a polytope



Improved basis for a polytope



Improved basis for a polytope



Improved basis for a polytope

For a given K, the critical determinant ($\Lambda \subset \mathbb{R}^n$) satisfies [Minkowski-Hlawka]

$$\Delta(K) \leq \mathrm{Vol}(K)$$

Scheme II

Using the **successive minima** of K we establish that there exists a **strictly admissible and integer lattice** such that

$$\Delta_{\mathbb{Z}}(K) \leq n!\ \mathrm{Vol}(K)$$

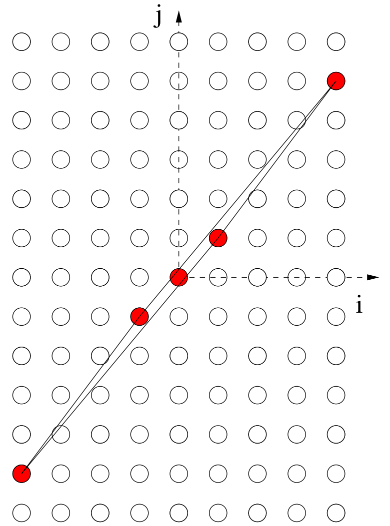
Guaranteed heuristics

$\det \Lambda \leq c_n \text{Vol}(K)$

Full dimensional polytope, arbitrary set in some cases

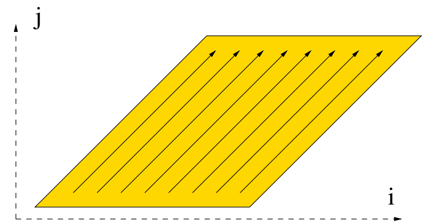
	Optimal linear
Enumeration, Λ such that $\det(\Lambda) \leq n!\text{Vol}(K)$	
Using the successive minima (Scheme II) (adapting [Rogers])	$\mathbf{c_n = n!}$
Based on K (Scheme I, $F_i(\vec{a}_i) \leq 1$)	$c_n = (n!)^2$
Generalized reduction (Scheme I)	$c_n = 2^{n^2} n!$
Based on K^* (Scheme I, $F_i^*(\vec{c}_i) \leq 1$) (cf [Lefebvre and Feautrier])	$c_n = (n!)^2$
Lenstra-Lenstra-Lovász reduction (ellipsoid approximation)	$c_n = 2^{n(n+3)/4} n^n$
$+$ 1D allocations, and power of two moduli	

Cf Limitations



Optimal size: 2
(unchanged)

New schedule: θ(i,j)=(i−j,i)



Previous heuristics: size $O(N)$ or $O(N^2)$

Guaranteed heuristics, $n = 2$:

$$\text{Size} = \det \Lambda \leq 2\ \text{Vol}(K) = 4.$$

Outline

In practice

Performance is guaranteed as soon as the **basis is appropriate** w.r.t K

- access functions to arrays are “simple”
- scheduling functions are not “too degenerated”
- writing domains are “not too skewed”

$\Rightarrow$ Mixing Lefebvre-Feautrier and Quilleré-Rajopadhye (schedule basis)

Computational aspects

Integer matrix manipulation for enumerative construction
Generalized basis reduction (Linear Programming)
Integer Linear Programming

Questions

Another approach for **obtaining integral and strictly admissible lattices**?

Power of linear allocations with respect to the **optimum**?

More general allocations, e.g. multi-periodic schemes?

More general conflicting indices set?