

# Fast Integer Programming in Fixed Dimension

Friedrich Eisenbrand

# Integer Programming

- ▶ **Variables:**  $x(1), \dots, x(n)$
- ▶ **Constraints:**  $a_{i1}x(1) + \dots + a_{in}x(n) \leq b(i)$ , for  $i = 1, \dots, m$
- ▶ **Linear objective function:**  $c(1)x(1) + \dots + c(n)x(n)$
  
- ▶ **Goal :** Find **integer assignment** to variables  $x(1), \dots, x(n)$ , such that constraints are **satisfied** and objective function value **maximized**

## IP in Fixed Dimension

- ▶ Integer programming is NP-complete (Karp 1972, Borosh & Treybig 1976)
- ▶ If **dimension is fixed**, then IP is **polynomially solvable** (Lenstra 1983)
  - ▶  $m$ : Number of constraints
  - ▶  $s$ : largest **binary encoding length** of coefficient in input
  - ▶ **Complexity** of Lenstra's algorithm:  $O(ms + s^2)$

## Why binary encoding length?

### Theorem

$$\gcd(a, b) = \min\{x a + y b \mid x, y \in \mathbb{Z}, x a + y b \geq 1\}$$

$$\begin{aligned} \text{minimize} \quad & x a + y b \\ & x a + y b \geq 1 \\ & x, y \in \mathbb{Z}. \end{aligned}$$

Integer Program with **two variables** and **one constraint**

# Why binary encoding length?

## Theorem

$$\gcd(a, b) = \min\{x a + y b \mid x, y \in \mathbb{Z}, x a + y b \geq 1\}$$

$$\begin{aligned} \text{minimize} \quad & x a + y b \\ & x a + y b \geq 1 \\ & x, y \in \mathbb{Z}. \end{aligned}$$

Integer Program with **two variables** and **one constraint**

Consequence:

IP in fixed dimension contains **gcd-computation**

# The Euclidean Algorithm

- ▶ Input: Integers  $a \geq b > 0$
- ▶ Output:  $\gcd(a, b)$
- ▶ **while**  $b \neq 0$ 
  - ▶ Compute  $q \geq 1$  and  $0 \leq r < b$  with  $a = qb + r$
  - ▶  $a \leftarrow b$
  - ▶  $b \leftarrow r$
- ▶ **return**  $a$

# The Euclidean Algorithm

- ▶ Input: Integers  $a \geq b > 0$
- ▶ Output:  $\gcd(a, b)$
- ▶ **while**  $b \neq 0$ 
  - ▶ Compute  $q \geq 1$  and  $0 \leq r < b$  with  $a = qb + r$
  - ▶  $a \leftarrow b$
  - ▶  $b \leftarrow r$
- ▶ **return**  $a$

## Analysis

- ▶  $r \leq a/2 \implies$  running time is  $O(\log a)$
- ▶ Binary encoding length is **upper** and **lower** bound on running time (number of arithmetic operations) (Knuth 1969)

# The Question

Is IP in **fixed dimension** and with a **fixed number of constraints** solvable in **linear time**?

( $O(s)$  number of arithmetic operations)

# The Question

Is IP in **fixed dimension** and with a **fixed number of constraints** solvable in **linear time**?

( $O(s)$  number of arithmetic operations)

**Yes** answer closes running time gap of IP to Euclidean algorithm

# A new Algorithm

- ▶  $m$ : Number of constraints
- ▶  $s$ : Bound on binary encoding length of coefficient

There exists an algorithm for IP with running time

- ▶  $O(s)$ , if  $m$  fixed  $O(s^2)$
- ▶ Expected  $O(m + s \log m)$  for varying  $m$   $O(m s + s^2)$

(E. 2003)

# Flatness

- ▶ **Polyhedron**: Rational points satisfying all constraints
- ▶ **Width** of  $P \subseteq \mathbb{R}^n$  along  $d$ ,  $w_d(P)$ :

$$\max\{d^T x \mid x \in P\} - \min\{d^T x \mid x \in P\}$$

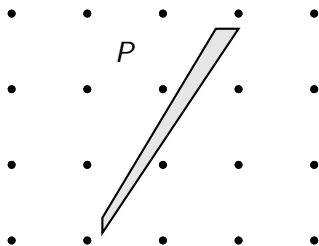
# Flatness

- ▶ **Polyhedron**: Rational points satisfying all constraints
- ▶ **Width** of  $P \subseteq \mathbb{R}^n$  along  $d$ ,  $w_d(P)$ :

$$\max\{d^T x \mid x \in P\} - \min\{d^T x \mid x \in P\}$$

## Theorem (Khinchine's Flatness Theorem)

If  $P \cap \mathbb{Z}^n = \emptyset$ , then there exists *integral*  $d \neq 0$  s.t. width of  $P$  along  $d$  bounded by constant  $f_n$



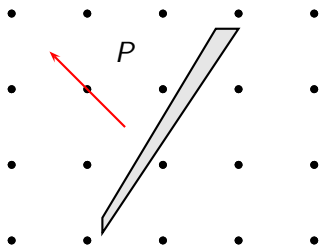
# Flatness

- ▶ **Polyhedron**: Rational points satisfying all constraints
- ▶ **Width** of  $P \subseteq \mathbb{R}^n$  along  $d$ ,  $w_d(P)$ :

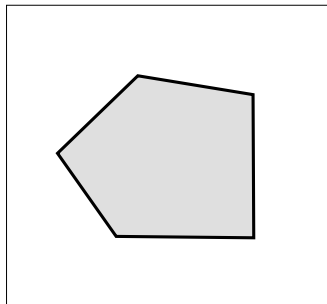
$$\max\{d^T x \mid x \in P\} - \min\{d^T x \mid x \in P\}$$

## Theorem (Khinchine's Flatness Theorem)

If  $P \cap \mathbb{Z}^n = \emptyset$ , then there exists *integral*  $d \neq 0$  s.t. width of  $P$  along  $d$  bounded by constant  $f_n$

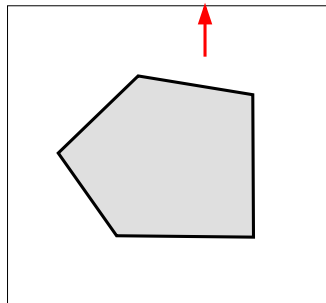


# Lenstra's Algorithm



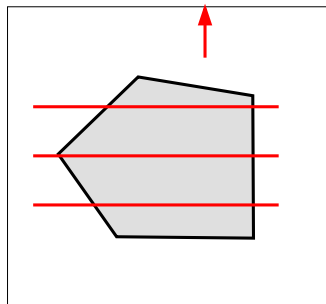
- ▶ Algorithm decides  $P \cap \mathbb{Z}^n = \emptyset$
- ▶ Compute **width of  $P$**  and corresponding integral direction  $d \in \mathbb{Z}^d$
- ▶ If width too large, then  $P \cap \mathbb{Z}^n \neq \emptyset$
- ▶ Otherwise search for integer point **recursively** on one of the hyperplanes  $(d^T x = \delta) \cap P$ ,  $\delta \in \mathbb{Z}$

# Lenstra's Algorithm



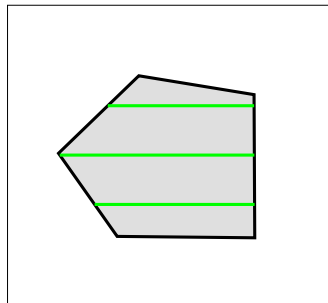
- ▶ Algorithm decides  $P \cap \mathbb{Z}^n = \emptyset$
- ▶ Compute **width of  $P$**  and corresponding integral direction  $d \in \mathbb{Z}^d$
- ▶ If width too large, then  $P \cap \mathbb{Z}^n \neq \emptyset$
- ▶ Otherwise search for integer point **recursively** on one of the hyperplanes  $(d^T x = \delta) \cap P$ ,  $\delta \in \mathbb{Z}$

# Lenstra's Algorithm



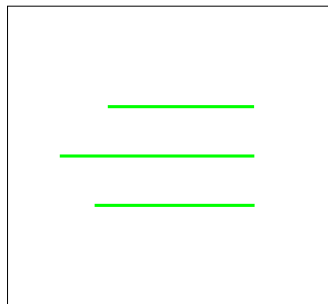
- ▶ Algorithm decides  $P \cap \mathbb{Z}^n = \emptyset$
- ▶ Compute **width of  $P$**  and corresponding integral direction  $d \in \mathbb{Z}^d$
- ▶ If width too large, then  $P \cap \mathbb{Z}^n \neq \emptyset$
- ▶ Otherwise search for integer point **recursively** on one of the hyperplanes  $(d^T x = \delta) \cap P$ ,  $\delta \in \mathbb{Z}$

# Lenstra's Algorithm



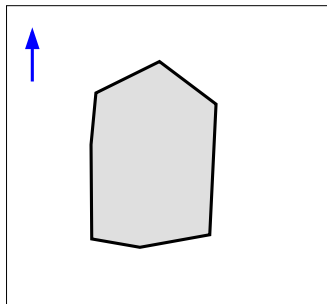
- ▶ Algorithm decides  $P \cap \mathbb{Z}^n = \emptyset$
- ▶ Compute **width of  $P$**  and corresponding integral direction  $d \in \mathbb{Z}^d$
- ▶ If width too large, then  $P \cap \mathbb{Z}^n \neq \emptyset$
- ▶ Otherwise search for integer point **recursively** on one of the hyperplanes  $(d^T x = \delta) \cap P$ ,  $\delta \in \mathbb{Z}$

# Lenstra's Algorithm



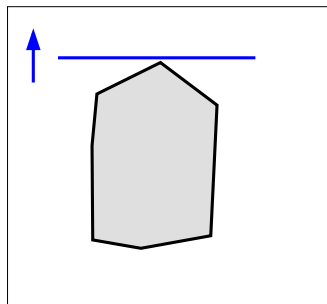
- ▶ Algorithm decides  $P \cap \mathbb{Z}^n = \emptyset$
- ▶ Compute **width of  $P$**  and corresponding integral direction  $d \in \mathbb{Z}^d$
- ▶ If width too large, then  $P \cap \mathbb{Z}^n \neq \emptyset$
- ▶ Otherwise search for integer point **recursively** on one of the hyperplanes  $(d^T x = \delta) \cap P$ ,  $\delta \in \mathbb{Z}$

## Idea: Sliding Objective Function



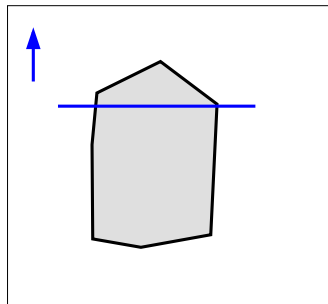
- ▶ Objective function **slides** into polyhedron
- ▶ Until upper part **too fat**
- ▶ **Optimum** lies on one of the constant number of hyperplanes
- ▶ **Recursively** search for **optimum** on each hyperplane

## Idea: Sliding Objective Function



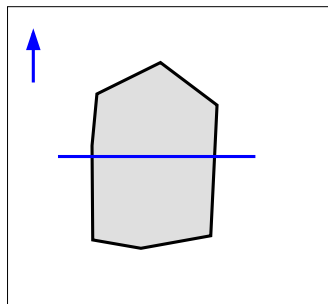
- ▶ Objective function **slides** into polyhedron
- ▶ Until upper part **too fat**
- ▶ **Optimum** lies on one of the constant number of hyperplanes
- ▶ **Recursively** search for **optimum** on each hyperplane

## Idea: Sliding Objective Function



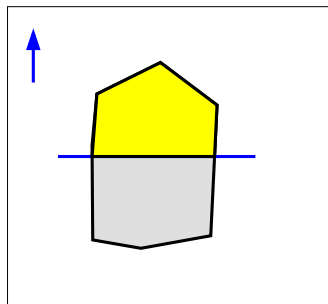
- ▶ Objective function **slides** into polyhedron
- ▶ Until upper part **too fat**
- ▶ **Optimum** lies on one of the constant number of hyperplanes
- ▶ **Recursively** search for **optimum** on each hyperplane

## Idea: Sliding Objective Function



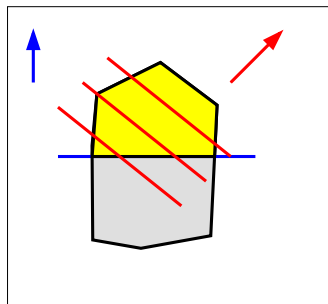
- ▶ Objective function **slides** into polyhedron
- ▶ Until upper part **too fat**
- ▶ **Optimum** lies on one of the constant number of hyperplanes
- ▶ **Recursively** search for **optimum** on each hyperplane

## Idea: Sliding Objective Function



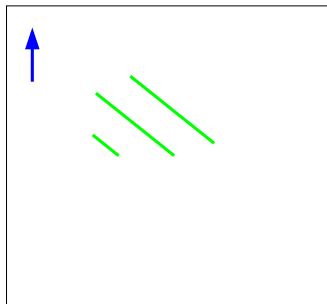
- ▶ Objective function **slides** into polyhedron
- ▶ Until upper part **too fat**
- ▶ **Optimum** lies on one of the constant number of hyperplanes
- ▶ **Recursively** search for **optimum** on each hyperplane

## Idea: Sliding Objective Function



- ▶ Objective function **slides** into polyhedron
- ▶ Until upper part **too fat**
- ▶ **Optimum** lies on one of the constant number of hyperplanes
- ▶ **Recursively** search for **optimum** on each hyperplane

## Idea: Sliding Objective Function



- ▶ Objective function **slides** into polyhedron
- ▶ Until upper part **too fat**
- ▶ **Optimum** lies on one of the constant number of hyperplanes
- ▶ **Recursively** search for **optimum** on each hyperplane

**Problem:** Geometry of truncated part changes

# The Width of a Simplex

▶  $\Sigma = \text{conv}\{0, v_1, \dots, v_n\}$  simplex

▶ Width of  $\Sigma$  along  $d$ :

$$w_d(\Sigma) = \max\{0, d^T v_1, \dots, d^T v_n\} - \min\{0, d^T v_1, \dots, d^T v_n\}$$

▶ A matrix with rows  $v_1^T, \dots, v_n^T$  then

$$\|A d\|_\infty \leq w_d(\Sigma) \leq 2 \|A d\|_\infty$$

# The Width of a Simplex

- ▶  $\Sigma = \text{conv}\{0, v_1, \dots, v_n\}$  simplex

- ▶ Width of  $\Sigma$  along  $d$ :

$$w_d(\Sigma) = \max\{0, d^T v_1, \dots, d^T v_n\} - \min\{0, d^T v_1, \dots, d^T v_n\}$$

- ▶ A matrix with rows  $v_1^T, \dots, v_n^T$  then

$$\|A d\|_\infty \leq w_d(\Sigma) \leq 2 \|A d\|_\infty$$

- ▶ Compute  $d \in \mathbb{Z}^n - \{0\}$  s.t.  $\|A d\|_\infty$  minimal

# The Width of a Simplex

- ▶  $\Sigma = \text{conv}\{0, v_1, \dots, v_n\}$  **simplex**

- ▶ Width of  $\Sigma$  along  $d$ :

$$w_d(\Sigma) = \max\{0, d^T v_1, \dots, d^T v_n\} - \min\{0, d^T v_1, \dots, d^T v_n\}$$

- ▶ A matrix with rows  $v_1^T, \dots, v_n^T$  then

$$\|A d\|_\infty \leq w_d(\Sigma) \leq 2 \|A d\|_\infty$$

- ▶ Compute  $d \in \mathbb{Z}^n - \{0\}$  s.t.  $\|A d\|_\infty$  **minimal**
- ▶ **Shortest vector** in lattice  $\Lambda(A) = \{A d \mid d \in \mathbb{Z}^n\}$

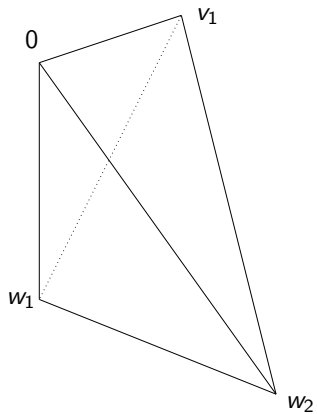
## Two-Layer Simplices

- ▶ Vertices can be partitioned into  $V$  and  $W$  such that

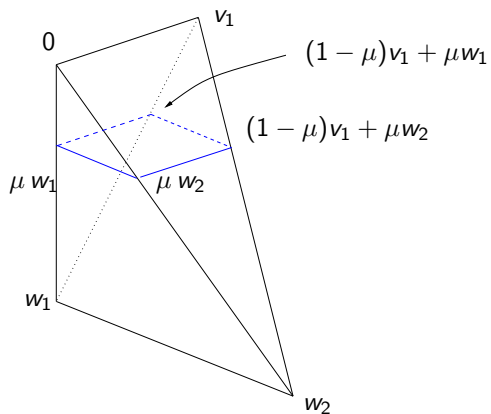
$$c^T v = c^T v' \text{ and } c^T w = c^T w' \text{ for all } v, v' \in V, w, w' \in W.$$

- ▶ Thus objective function value is equal for points in  $V$  and  $W$  respectively
- ▶ Polytope with **fixed number of facets** can be covered by **fixed number** of two-layer simplices

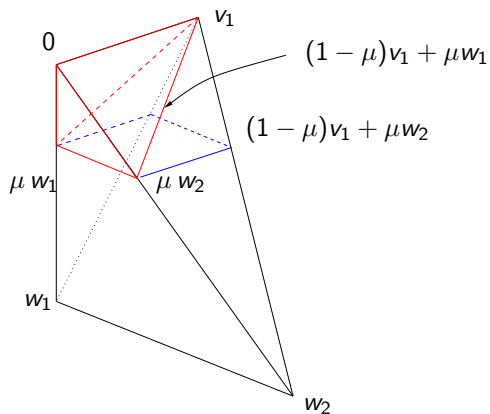
## Example in 3D



## Example in 3D



## Example in 3D



Width of upper part is about width of red simplex, spanned by  $V$  and  $\mu W$ .

## Reduction to Parametric Shortest Vector

Computation of  $\mu$  such that truncated part exceeds flatness constant reduces to:

**PARAMETRIC SHORTEST VECTOR:**

Given  $A \in \mathbb{Z}^{n \times n}$  and parameter  $U \in \mathbb{N}$ , find parameter  $\mu \in \mathbb{N}$   
s.t.  $U \leq \text{SV}(\Lambda(A_{\mu,k})) \leq 2^{n+1/2} \cdot U$

## Algorithm for PSV

```
 $p \leftarrow 2^{\lceil \log U \rceil}$   
 $B \leftarrow A_{p,k}$   
repeat  
  if  $p = 1$   
    return  $\text{SV}(\Lambda) > U$   
   $B \leftarrow B_{1/2,k}$   
   $p \leftarrow p/2$   
   $B \leftarrow \text{LLL}(B)$   
until  $\|b_1\| \leq 2^{(n-1)/2} \cdot U$   
return  $2p$ 
```

# Running Time

- ▶ **Potential** of basis:  $\phi(B) = \|b_1^*\|^{2n} \|b_2^*\|^{2(n-1)} \dots \|b_n^*\|^2$
- ▶ Potential is strictly decreasing
- ▶  $B_1 \rightarrow \text{LLL} \rightarrow B_2$ :  $\log \phi(B_1) - \log \phi(B_2)$  iterations
- ▶  $\phi(A_{U,k}) \leq \phi(A_{U,n}) \leq U^{2n^2} (\|a_1\| \dots \|a_n\|)^{2n}$
- ▶ In fixed dimension:  $O(\text{size}(U) + \text{size}(A))$  iterations, **linear**

## Theorem (E. 2003)

*Parametric shortest vector can be solved in **linear** time, if dimension is fixed.*

# Complexity of Integer Programming

- ▶  $m$ : Number of constraints
- ▶  $s$ : Max. binary encoding-length of coefficient

## Theorem (E. 2003)

*There exists an algorithm for IP with running time*

- $O(s)$ , if  $m$  fixed*
- Expected  $O(m + s \log m)$  for varying  $m$  (with Clarkson's random sampling algorithm)*