

Algorithms for integer programming in fixed dimension

Friedrich Eisenbrand

Problem I

Is there an $O(m + s)$ algorithm for integer programming in fixed dimension?

Problem I: Outline

- **Combinatorics & Number Theory:** The two sides of the medal
- Algorithms for linear programming in fixed dimension.
 - Clarkson's algorithm
 - IP is a problem of LP-type
- Linear time algorithm for IP and fixed number of constraints
 - Lattices and basis reduction
 - The Flatness theorem and integer feasibility
 - The LLL-algorithm
 - Avoiding binary search to find optimum solution
- Combining techniques: Randomized $O(m + s \cdot \log m)$ algorithm for IP

GCDs and IP

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

minimize $xa + yb$

condition $xa + yb \geq 1$

$x, y \in \mathbb{Z}.$

GCDs and IP

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

minimize $xa + yb$

condition $xa + yb \geq 1$

$x, y \in \mathbb{Z}.$

Integer Programming: **Combinatorics & Number Theory**

Euclidean Algorithm

- Input: Integers $a \geq b > 0$
- 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

Euclidean Algorithm

- Input: Integers $a \geq b > 0$
- 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 $O(\log a)$
- Running time depends on binary encoding length of numbers

Complexity of IP

Complexity measure:

- **Arithmetic model:** Count number of arithmetic operations
- **Size of numbers:** Encoding length of numbers in course of algorithm remains small

m : Number of constraints

s : Largest binary encoding length of number in input

- Lenstra's Algorithm:
 - $O(m + s)$ for **feasibility**
 - $O(s \cdot (m + s))$ for **optimization**
- Euclidean algorithm: $O(s)$
- **Goal:** $O(m + s)$
- **Achieved so far:** $O(m + s \cdot \log m)$

Linear Programming

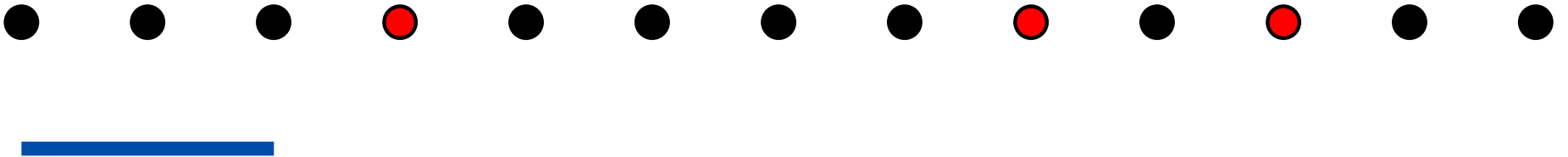
Quiz

- $H = \{1, \dots, n\}$, $r \in H$, $R \in \binom{H}{r}$ drawn uniformly at random
- $V_R = \min\{i \in R\} - 1$
- What is $E[V_R]$?



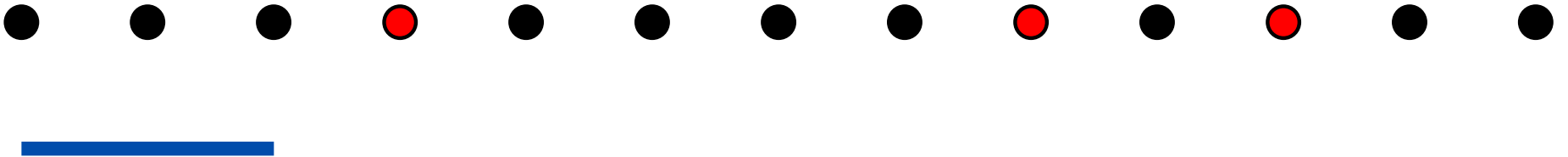
Quiz

- $H = \{1, \dots, n\}$, $r \in H$, $R \in \binom{H}{r}$ drawn uniformly at random
- $V_R = \min\{i \in R\} - 1$
- What is $E[V_R]$?



Quiz

- $H = \{1, \dots, n\}$, $r \in H$, $R \in \binom{H}{r}$ drawn uniformly at random
- $V_R = \min\{i \in R\} - 1$
- What is $E[V_R]$?



Answer: $(n - r) / (r + 1)$

Proof

- For $Q \subseteq H$ and $j \in H$ define $\chi(Q, j) = \begin{cases} 1 & \text{if } j < \min\{i \in Q\}, \\ 0 & \text{otherwise.} \end{cases}$
- $E[V_R] = \left(\sum_{R \in \binom{H}{r}} \sum_{j \in H \setminus R} \chi(R, j) \right) / \binom{n}{r}$
- One has $\binom{n}{r} \cdot (n - r) = \binom{n}{r+1} \cdot (r + 1)$
- Thus

$$\begin{aligned} \binom{n}{r} \cdot E[V_R] &= \sum_{Q \in \binom{H}{r+1}} \sum_{j \in Q} \chi(Q - \{j\}, j) \\ &= \binom{n}{r+1}. \end{aligned}$$

- Thus $E[V_R] = \binom{n}{r+1} / \binom{n}{r} = (n - r) / (r + 1)$

Linear Programming

- Given: Set H of m linear constraints in \mathbb{R}^d and $H^- = \{x(i) \leq M \mid i = 1, \dots, d\}$ explicit upper bounds
- For $G \subseteq H$, $x^*(G)$ is **lex. max.** point satisfying all $h \in G \cup H^-$
- Task: **Compute** $x^*(H)$

Linear Programming

- Given: Set H of m linear constraints in \mathbb{R}^d and $H^- = \{x(i) \leq M \mid i = 1, \dots, d\}$ explicit upper bounds
- For $G \subseteq H$, $x^*(G)$ is **lex. max.** point satisfying all $h \in G \cup H^-$
- Task: **Compute** $x^*(H)$

Quiz:

- Choose $R \in \binom{H}{r}$ uniformly at random
- $V_R = \{h \in H \mid x^*(R) \text{ violates } h\}$
- **What is** $E[|V_R|]$?

Linear Programming

- Given: Set H of m linear constraints in \mathbb{R}^d and $H^- = \{x(i) \leq M \mid i = 1, \dots, d\}$ explicit upper bounds
- For $G \subseteq H$, $x^*(G)$ is **lex. max.** point satisfying all $h \in G \cup H^-$
- Task: **Compute** $x^*(H)$

Quiz:

- Choose $R \in \binom{H}{r}$ uniformly at random
- $V_R = \{h \in H \mid x^*(R) \text{ violates } h\}$
- **What is** $E[|V_R|]$?

Answer: at most $(m - r)/(r + 1) \cdot d$

Proof

- $E[|V_R|] = \left(\sum_{R \in \binom{H}{r}} |V_R| \right) / \binom{m}{r}$
- For $Q \subseteq H$ and $h \in H$ define
$$\chi_G(Q, h) = \begin{cases} 1 & \text{if } x^*(Q) \text{ violates } h, \\ 0 & \text{otherwise.} \end{cases}$$

$$\begin{aligned} \binom{m}{r} E(|V_R|) &= \sum_{R \in \binom{H}{r}} \sum_{h \in H \setminus R} \chi_G(R, h) \\ &= \sum_{Q \in \binom{H}{r+1}} \sum_{h \in Q} \chi_G(Q - h, h) \\ &\leq \sum_{Q \in \binom{H}{r+1}} d \\ &= \binom{m}{r+1} \cdot d. \end{aligned}$$

Sampling Lemma

Lemma (Gärtner & Welzl 1996). *Let G and H (multi-)sets of constraints $|H| = m$ and let $1 \leq r \leq m$. Then for random $R \in \binom{H}{r}$:*

$$E[|V_R|] \leq d(m - r)/(r + 1),$$

where $V_R = \{h \in H \mid x^*(G \cup R) \text{ violates } h\}$.

Sampling Lemma

Lemma (Gärtner & Welzl 1996). *Let G and H (multi-)sets of constraints $|H| = m$ and let $1 \leq r \leq m$. Then for random $R \in \binom{H}{r}$:*

$$E[|V_R|] \leq d(m - r)/(r + 1),$$

where $V_R = \{h \in H \mid x^*(G \cup R) \text{ violates } h\}$.

Set $r = \lceil d \cdot \sqrt{m} \rceil$ then

$$E[|V_R|] \leq d \cdot (m - r)/(r + 1) \leq Dm/r \leq \sqrt{m}.$$

Clarkson's algorithm I

1. Input: H with $|H| = m$
2. $r \leftarrow d \cdot \sqrt{m}$
3. $G \leftarrow \emptyset$
4. REPEAT
 - (a) Choose random $R \in \binom{H}{r}$
 - (b) Compute $x^* = x^*(G \cup R)$
 - (c) $V_R \leftarrow \{h \in H \mid x^* \text{ violates } h\}$
 - (d) IF $|V_R| \leq 2\sqrt{m}$
THEN $G \leftarrow G \cup V_R,$
5. UNTIL $V_R = \emptyset$

Clarkson's algorithm I

1. Input: H with $|H| = m$
2. $r \leftarrow d \cdot \sqrt{m}$
3. $G \leftarrow \emptyset$
4. REPEAT
 - (a) Choose random $R \in \binom{H}{r}$
 - (b) Compute $x^* = x^*(G \cup R)$
 - (c) $V_R \leftarrow \{h \in H \mid x^* \text{ violates } h\}$
 - (d) IF $|V_R| \leq 2\sqrt{m}$
THEN $G \leftarrow G \cup V_R$,
5. UNTIL $V_R = \emptyset$

Sample size

Clarkson's algorithm I

1. Input: H with $|H| = m$

2. $r \leftarrow d \cdot \sqrt{m}$

Sample size

3. $G \leftarrow \emptyset$

Contains optimal basis in the end

4. REPEAT

(a) Choose random $R \in \binom{H}{r}$

(b) Compute $x^* = x^*(G \cup R)$

(c) $V_R \leftarrow \{h \in H \mid x^* \text{ violates } h\}$

(d) IF $|V_R| \leq 2\sqrt{m}$
THEN $G \leftarrow G \cup V_R,$

5. UNTIL $V_R = \emptyset$

Clarkson's algorithm I

1. Input: H with $|H| = m$
2. $r \leftarrow d \cdot \sqrt{m}$ Sample size
3. $G \leftarrow \emptyset$ Contains optimal basis in the end
4. REPEAT
 - (a) Choose random $R \in \binom{H}{r}$
 - (b) Compute $x^* = x^*(G \cup R)$ with some other algorithm
 - (c) $V_R \leftarrow \{h \in H \mid x^* \text{ violates } h\}$
 - (d) IF $|V_R| \leq 2\sqrt{m}$
THEN $G \leftarrow G \cup V_R,$
5. UNTIL $V_R = \emptyset$

Clarkson's algorithm I

1. Input: H with $|H| = m$
2. $r \leftarrow d \cdot \sqrt{m}$ Sample size
3. $G \leftarrow \emptyset$ Contains optimal basis in the end
4. REPEAT
 - (a) Choose random $R \in \binom{H}{r}$
 - (b) Compute $x^* = x^*(G \cup R)$ with some other algorithm
 - (c) $V_R \leftarrow \{h \in H \mid x^* \text{ violates } h\}$
 - (d) IF $|V_R| \leq 2\sqrt{m}$ With probability $\geq 1/2$ true
THEN $G \leftarrow G \cup V_R,$
5. UNTIL $V_R = \emptyset$

Clarkson's algorithm I

1. Input: H with $|H| = m$
2. $r \leftarrow d \cdot \sqrt{m}$ Sample size
3. $G \leftarrow \emptyset$ Contains optimal basis in the end
4. REPEAT
 - (a) Choose random $R \in \binom{H}{r}$
 - (b) Compute $x^* = x^*(G \cup R)$ with some other algorithm
 - (c) $V_R \leftarrow \{h \in H \mid x^* \text{ violates } h\}$
 - (d) IF $|V_R| \leq 2\sqrt{m}$
THEN $G \leftarrow G \cup V_R,$ With probability $\geq 1/2$ true
successful iteration
5. UNTIL $V_R = \emptyset$

Clarkson's algorithm I

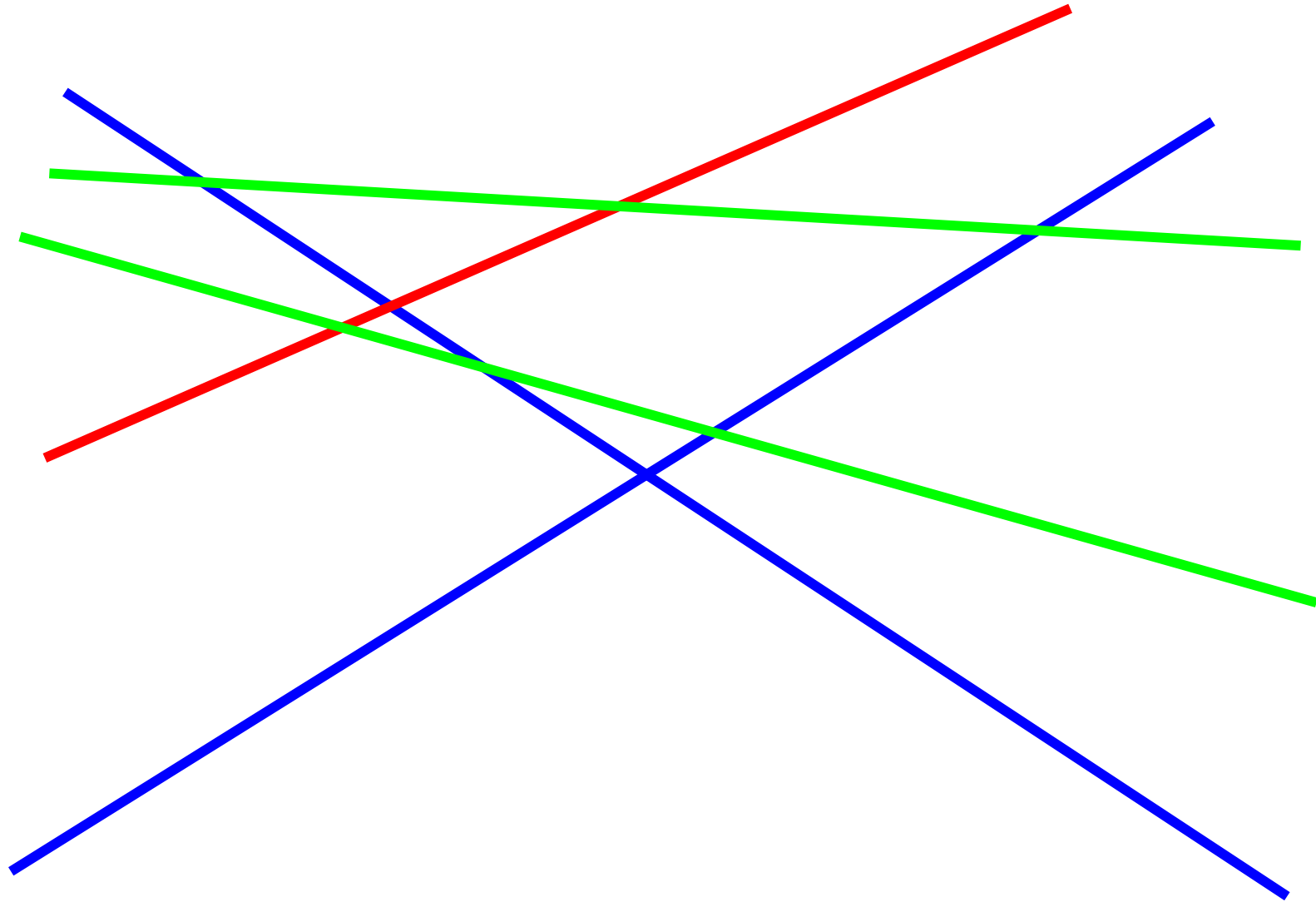
1. Input: H with $|H| = m$
2. $r \leftarrow d \cdot \sqrt{m}$ Sample size
3. $G \leftarrow \emptyset$ Contains optimal basis in the end
4. REPEAT
 - (a) Choose random $R \in \binom{H}{r}$
 - (b) Compute $x^* = x^*(G \cup R)$ with some other algorithm
 - (c) $V_R \leftarrow \{h \in H \mid x^* \text{ violates } h\}$
 - (d) IF $|V_R| \leq 2\sqrt{m}$ With probability $\geq 1/2$ true
THEN $G \leftarrow G \cup V_R,$ successful iteration
5. UNTIL $V_R = \emptyset$

At most d successful iterations

Invariant: G contains at most $2 \cdot d \cdot \sqrt{m}$ constraints

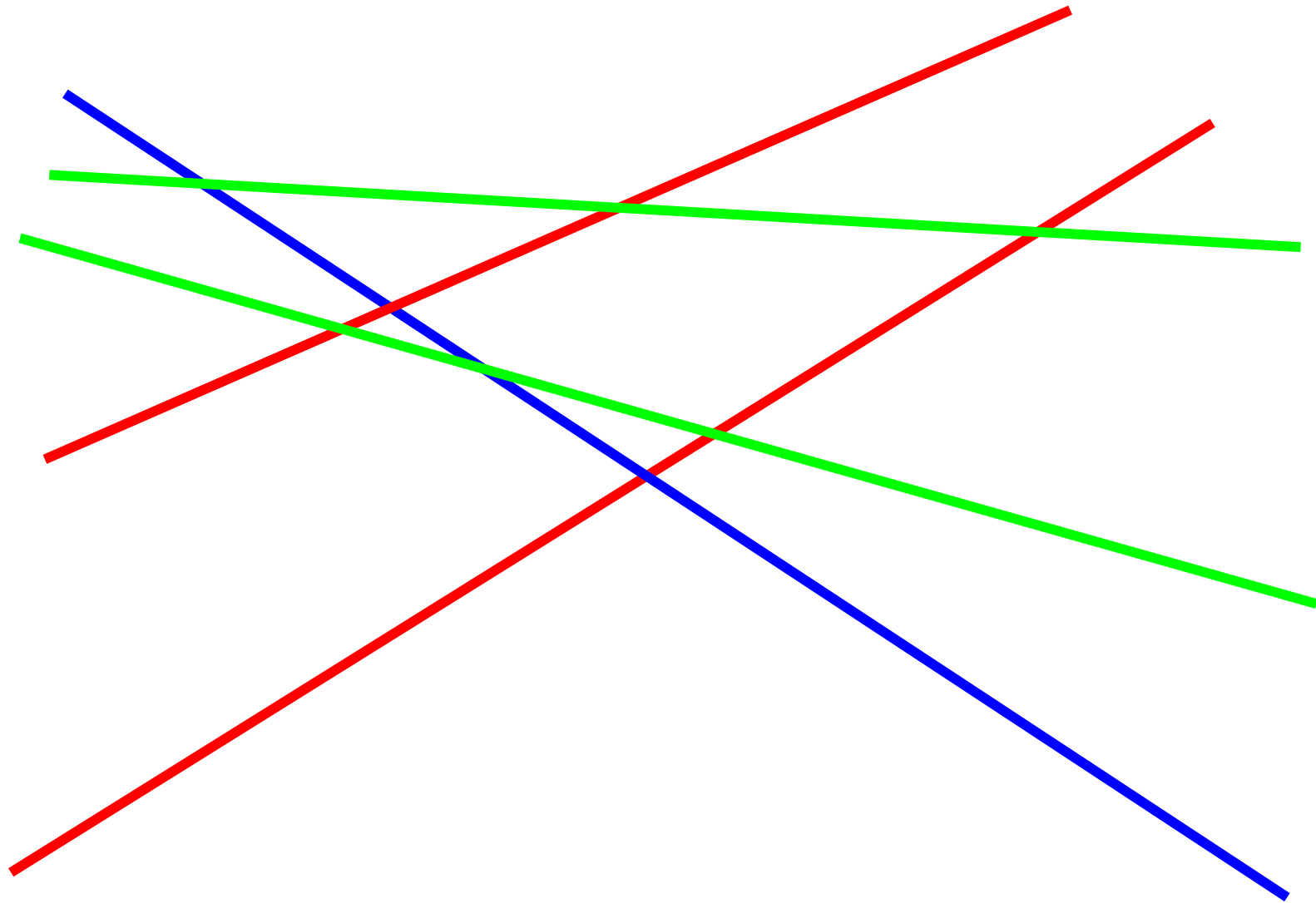
Example

R, G, B



Example

R, G, B



Analysis

In Step (4.c): $E[|V|] \leq \sqrt{m}$.

Let B be optimal basis.

- Each successful iteration, a **new** element of B enters G
- Thus at most d succ. it.
- $P(|V_R| > 2\sqrt{m}) \leq 1/2$ Markow inequality
- Expected number of iterations is $2d$

Clarkson 1 performs:

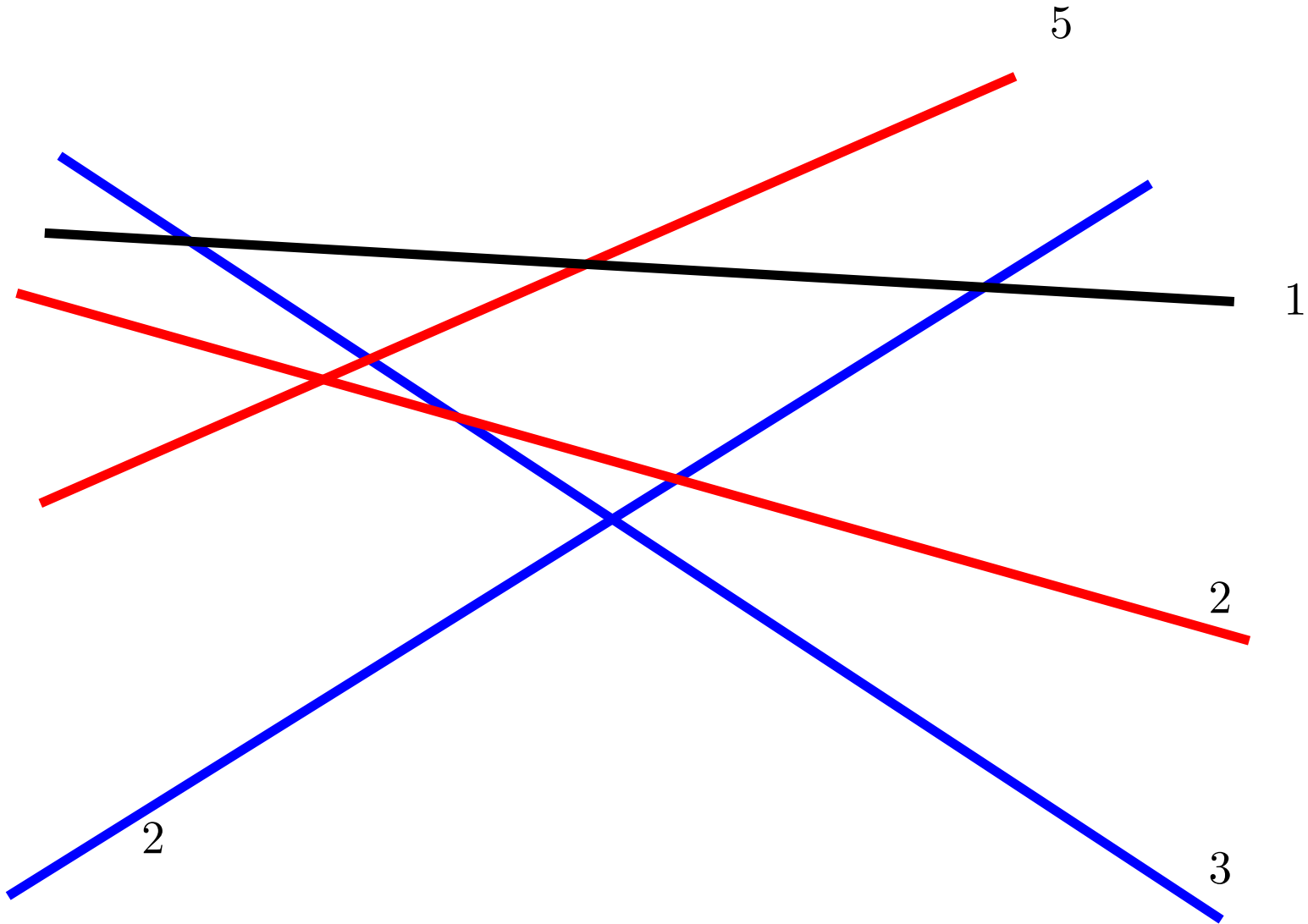
- Expected $2d$ calls to linear programming **oracle** with at most $3 \cdot d\sqrt{m}$ constraints
- Expected number of $O(d^2 \cdot m)$ **arithmetic operations**

Clarkson's algorithm II

- Each $h \in H$ is assigned a **multiplicity** μ_h .
- In the beginning $\mu_h = 1$ for all $h \in H$.
- Sample size r is small
- Idea: If $x^*(R)$ violates h , then **multiplicity/probability** is doubled
- Constraints of optimum basis become much more likely to be drawn next time
- We stop if R contains optimum basis

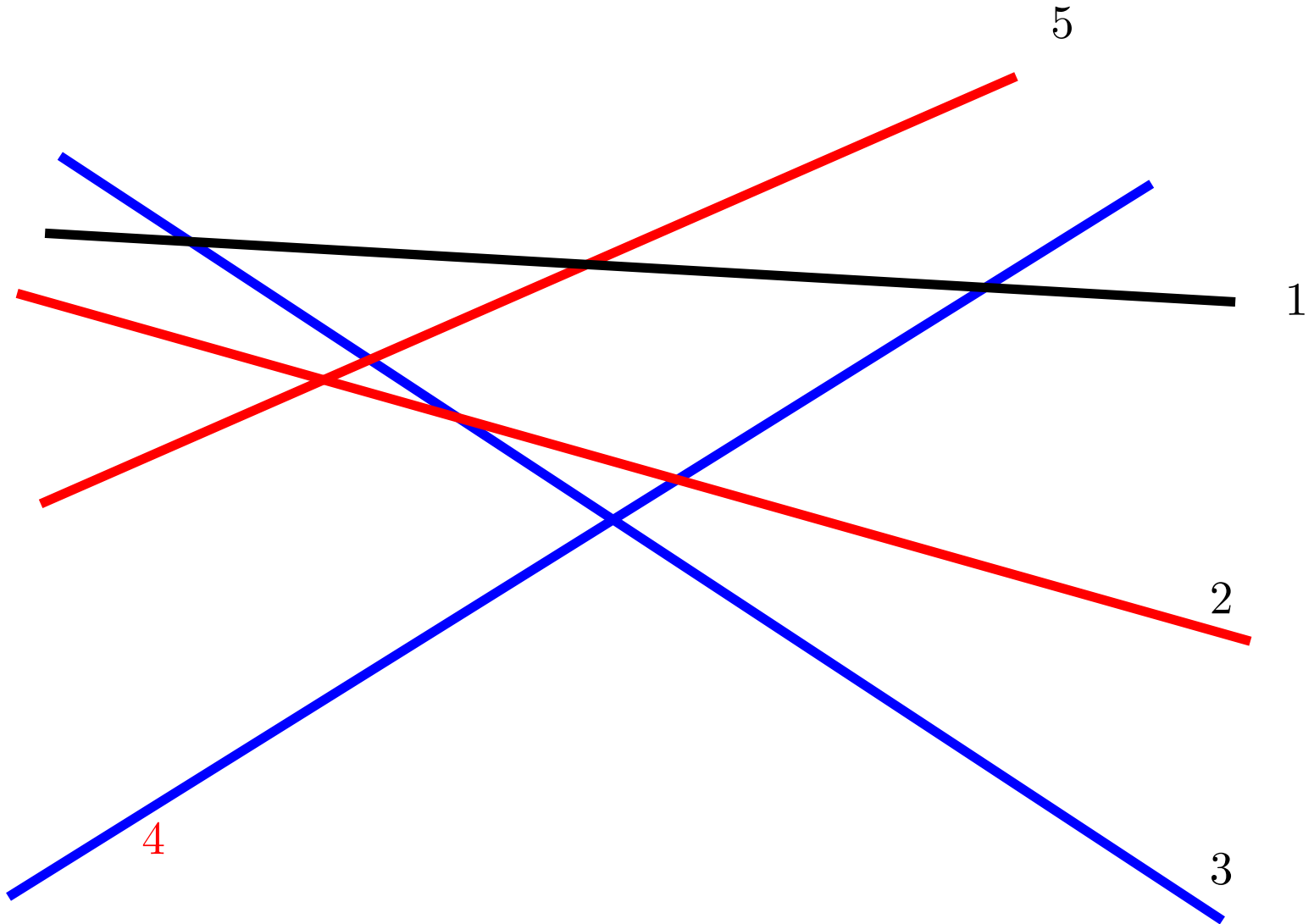
Example

R, B



Example

R, B



Clarkson 2

1. INPUT: H , $|H| = m$
2. $r \leftarrow 6 \cdot d^2$
3. REPEAT:
 - (a) Choose random $R \in \binom{H}{r}$
 - (b) Compute $x^* = x^*(R)$,
 - (c) $V_R \leftarrow \{h \in H \mid x^* \text{ violates } h\}$
 - (d) IF $\mu(V_R) \leq 1/(3d)\mu(H)$
THEN for all $h \in V$ do $\mu_h \leftarrow 2\mu_h$
4. UNTIL $V_R = \emptyset$

Clarkson 2

1. INPUT: H , $|H| = m$
2. $r \leftarrow 6 \cdot d^2$
3. REPEAT:
 - (a) Choose random $R \in \binom{H}{r}$
 - (b) Compute $x^* = x^*(R)$,
 - (c) $V_R \leftarrow \{h \in H \mid x^* \text{ violates } h\}$
 - (d) IF $\mu(V_R) \leq 1/(3d)\mu(H)$
THEN for all $h \in V$ do $\mu_h \leftarrow 2\mu_h$
4. UNTIL $V_R = \emptyset$

sample size

Clarkson 2

1. INPUT: H , $|H| = m$
2. $r \leftarrow 6 \cdot d^2$ sample size
3. REPEAT:
 - (a) Choose random $R \in \binom{H}{r}$ will contain optimum basis
 - (b) Compute $x^* = x^*(R)$,
 - (c) $V_R \leftarrow \{h \in H \mid x^* \text{ violates } h\}$
 - (d) IF $\mu(V_R) \leq 1/(3d)\mu(H)$
THEN for all $h \in V$ do $\mu_h \leftarrow 2\mu_h$
4. UNTIL $V_R = \emptyset$

Clarkson 2

1. INPUT: H , $|H| = m$
2. $r \leftarrow 6 \cdot d^2$ sample size
3. REPEAT:
 - (a) Choose random $R \in \binom{H}{r}$ will contain optimum basis
 - (b) Compute $x^* = x^*(R)$, with some other algorithm
 - (c) $V_R \leftarrow \{h \in H \mid x^* \text{ violates } h\}$
 - (d) IF $\mu(V_R) \leq 1/(3d)\mu(H)$
THEN for all $h \in V$ do $\mu_h \leftarrow 2\mu_h$
4. UNTIL $V_R = \emptyset$

Clarkson 2

1. INPUT: H , $|H| = m$
2. $r \leftarrow 6 \cdot d^2$ sample size
3. REPEAT:
 - (a) Choose random $R \in \binom{H}{r}$ will contain optimum basis
 - (b) Compute $x^* = x^*(R)$, with some other algorithm
 - (c) $V_R \leftarrow \{h \in H \mid x^* \text{ violates } h\}$
 - (d) IF $\mu(V_R) \leq 1/(3d)\mu(H)$ probability $\geq 1/2$
THEN for all $h \in V$ do $\mu_h \leftarrow 2\mu_h$
4. UNTIL $V_R = \emptyset$

Clarkson 2

1. INPUT: H , $|H| = m$
2. $r \leftarrow 6 \cdot d^2$ sample size
3. REPEAT:
 - (a) Choose random $R \in \binom{H}{r}$ will contain optimum basis
 - (b) Compute $x^* = x^*(R)$, with some other algorithm
 - (c) $V_R \leftarrow \{h \in H \mid x^* \text{ violates } h\}$
 - (d) IF $\mu(V_R) \leq 1/(3d)\mu(H)$ probability $\geq 1/2$
THEN for all $h \in V$ do $\mu_h \leftarrow 2\mu_h$ re-weighting
4. UNTIL $V_R = \emptyset$

Lemma. *B optimal basis, after kd successful iterations (entering re-weighting step):*

$$2^k \leq \mu(B) \leq m e^{k/3}, \text{ for basis } B \text{ of } H.$$

Proof:

- After $k \cdot d$ iterations: $\mu(B) \geq 2^k$
- Also $\mu(B) \leq \mu(H)$ and
 - After re-weighting:
$$\mu(H) \leq \mu_{old}(H) + 1/(3d) \cdot \mu(H) = (1 + 1/(3d))\mu_{old}(H)$$
 - Initially $\mu(H) = m$
 - Thus $\mu(H) \leq m \cdot (1 + 1/(3d))^{k \cdot d} \leq m \cdot e^{k/3}$

Complexity Clarkson 2

- $2^k \leq me^{k/3}$ implies $k \in O(\log m)$
- Expected number of $O(d \cdot \log m)$ iterations

Clarkson 2 requires

- expected number of $O(d^2 m \log m)$ arithmetic operations
- expected $6d \ln m$ base cases with $6 \cdot d^2$ constraints

Combining Clarkson 1 and 2

- $O(d^2 \cdot m)$ arithmetic operations
- $2 \cdot d$ calls to Clarkson 2 on $O(d\sqrt{m})$ constraints
 - $O(d^2\sqrt{m}\log m)$ arithmetic operations
 - $O(d\log m)$ calls to LP-oracle with $6 \cdot d^2$ constraints

Linear program can be solved

- with expected $O(d^3 \cdot m)$ arithmetic operations
- and $O(d^2 \cdot \log m)$ oracle calls to solve an LP with $6 \cdot d^2$ constraints
- in linear time if d is fixed

(Clarkson 1995)

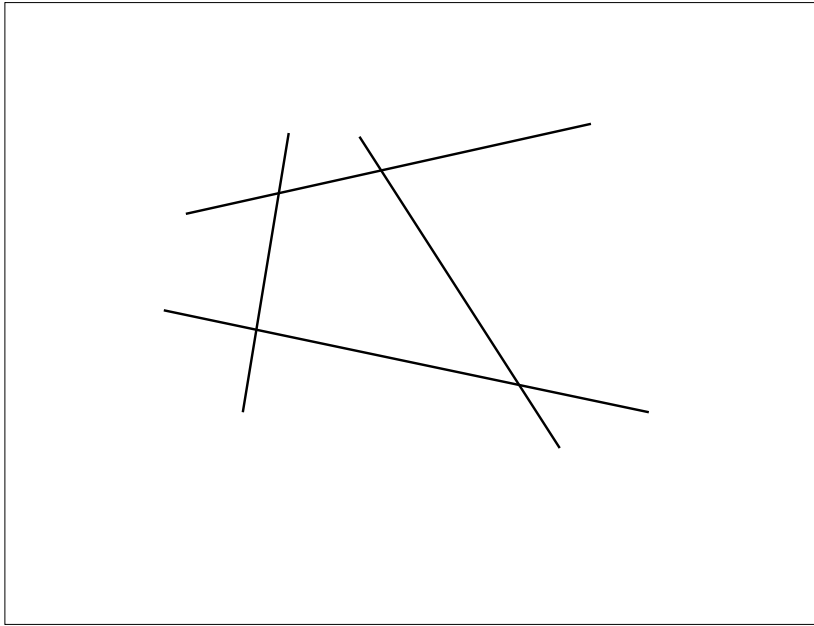
Integer Programming

- Given set H of m **integral constraints** in dimension d and $H^- = \{x(i) \leq M \mid i = 1, \dots, d\}$ explicit bound constraints.
- For $G \subseteq H$, $x^*(G)$ is lex. max. **integer point** satisfying G and H^- .
- **Task:** Compute $x^*(H)$.

A theorem of Bell and Scarf

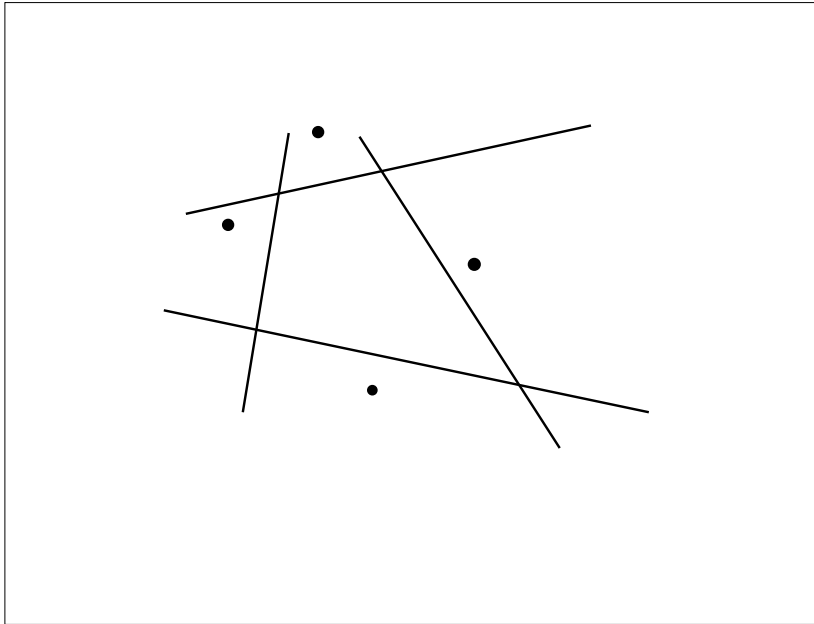
Theorem. *Let H be a set of rational linear constraints in \mathbb{R}^d . If there does not exist an integer point which satisfies all constraints, then there exists a subset $B \subseteq H$ with $|B| \leq 2^d$ such that there does not exist an integer point which satisfies all constraints in B .*

Proof



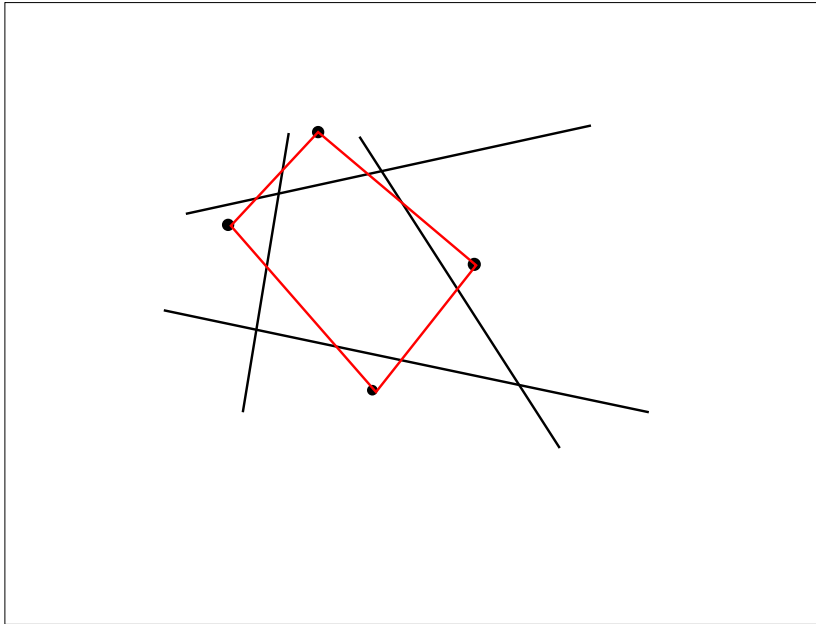
- Let H be minimal such that H has no feasible integer point, $m = |H| > 2^d$
- Assume constraints are $a_i^T x \leq \beta_i$ $i = 1, \dots, m$, where a_i and β_i are integers

Proof



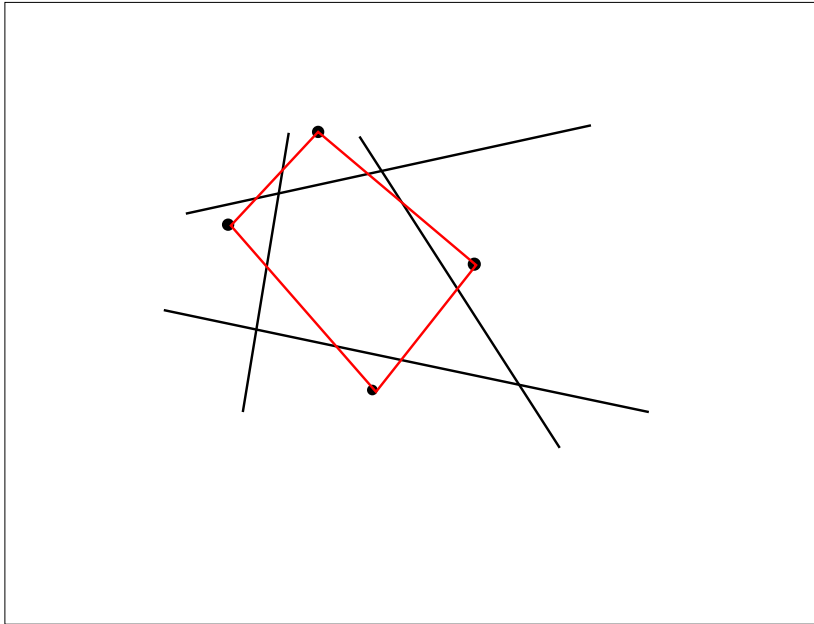
- Let H be minimal such that H has no feasible integer point, $m = |H| > 2^d$
- Assume constraints are $a_i^T x \leq \beta_i$ $i = 1, \dots, m$, where a_i and β_i are **integers**
- For each $a_i^T x \leq \beta_i$, there exists integer solution y_i which satisfies **all but the i -th** constraint.

Proof



- Let H be minimal such that H has no feasible integer point, $m = |H| > 2^d$
- Assume constraints are $a_i^T x \leq \beta_i$ $i = 1, \dots, m$, where a_i and β_i are integers
- For each $a_i^T x \leq \beta_i$, there exists integer solution y_i which satisfies all but the i -th constraint.
- $Z = \text{conv}(\{y_1, \dots, y_m\} \cap \mathbb{Z}^n)$

Proof



- Let $\gamma_1, \dots, \gamma_m \in \mathbb{Z}$ s.t. $\beta_i \leq \gamma_i$, system $a_i^T x \leq \gamma_i, i = 1, \dots, m$ has no solution in Z and $\gamma_1 + \dots + \gamma_m$ is maximal
- For each i there exists a $z_i \in Z$ s.t. $a_i^T z_i = \gamma_i + 1$ and $a_j^T z_i \leq \gamma_j$ for each $j \neq i$
- Since $m > 2^n$ there exist $i \neq j$ with $z_i \equiv z_j \pmod{2} \implies 1/2(z_i + z_j) \in Z$ and satisfies all constraints which is a contradiction

Exercise

Prove the following theorem

Theorem. *Let H be a set of linear constraints. If $x^*(H)$ exists then there exists a subset B of H with $|B| \leq 2^d - 1$ with $x^*(H) = x^*(B)$.*

- This B is called a **basis** of H .
- $D = 2^n - 1$ is **combinatorial dimension**

Complexity of IP

- Apply Clarkson's algorithm
- IP with m constraints in fixed dimension can be solved with $O(m)$ arithmetic operations and $O(\log m)$ oracle calls to solve IP with **fixed number of constraints**.

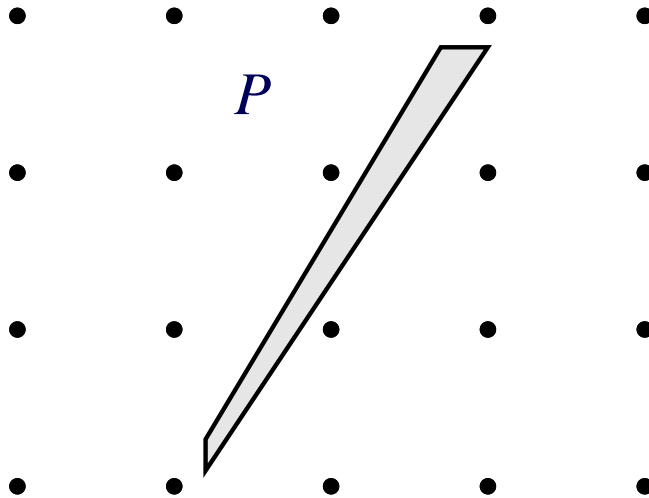
**Solving IP with fixed number of
constraints in linear time**

Integer feasibility

Flatness theorem

Width of P along c , $w_c(P)$: $\max\{c^T x \mid x \in P\} - \min\{c^T x \mid x \in P\}$

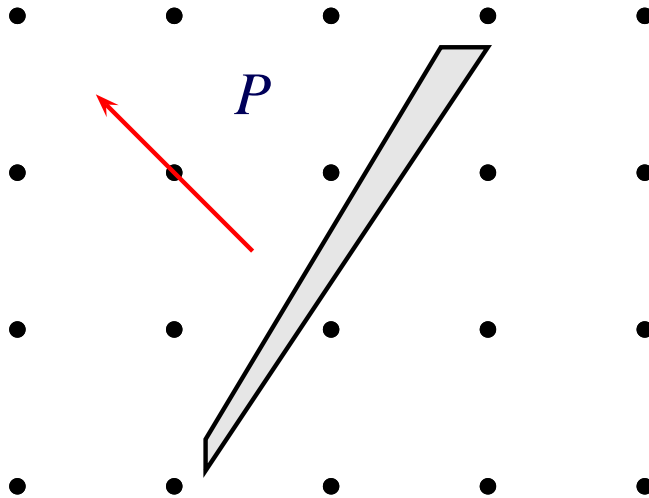
Theorem (Khinchine's flatness theorem). *If $P_I = \emptyset$, then there exists integral $c \neq 0$ such that width of P along c is \leq constant f_n*



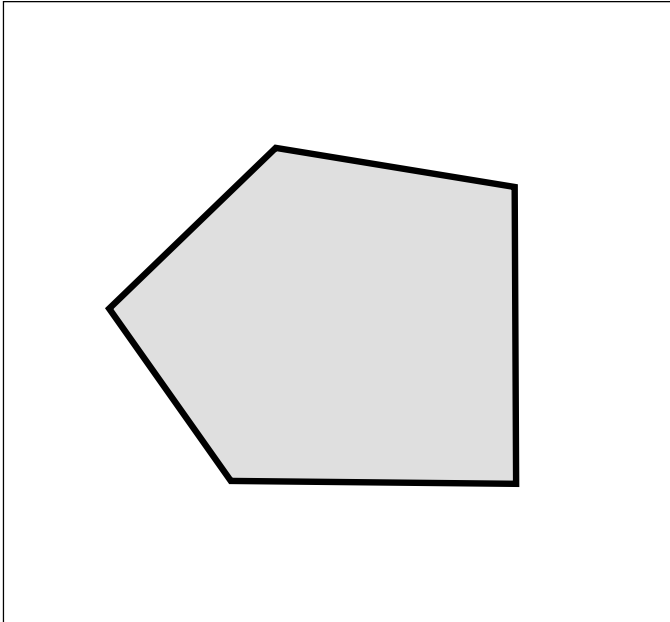
Flatness theorem

Width of P along c , $w_c(P)$: $\max\{c^T x \mid x \in P\} - \min\{c^T x \mid x \in P\}$

Theorem (Khinchine's flatness theorem). *If $P_I = \emptyset$, then there exists integral $c \neq 0$ such that width of P along c is \leq constant f_n*

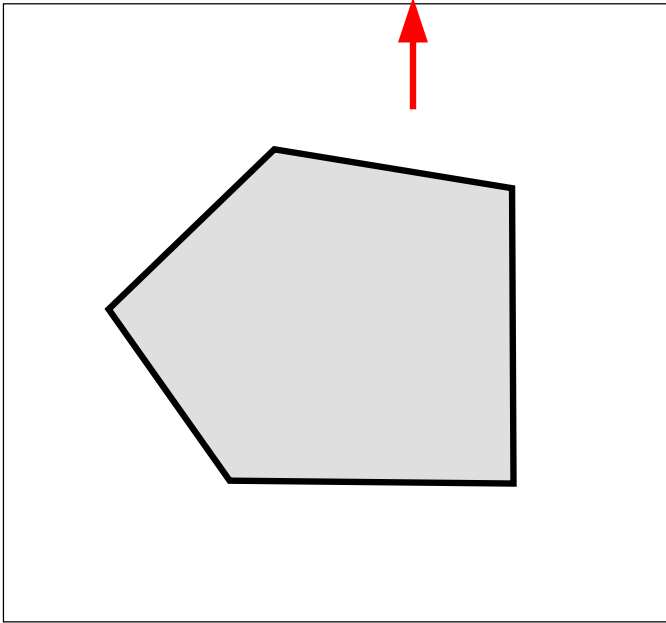


Lenstra's IP algorithm



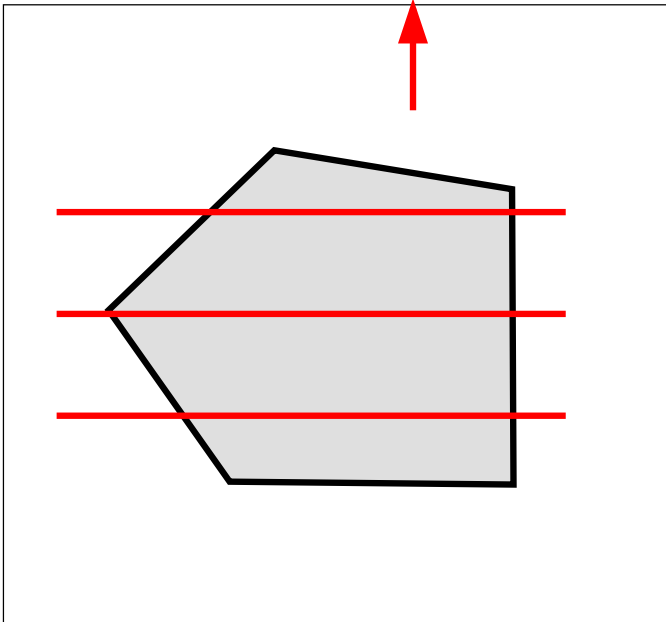
- Lenstra's algorithm is an algorithm for IP feasibility
- Computes width of polyhedron
- If width is too large, then return **feasible**
- Otherwise, **recursively** search for integer point on one of the constant number of hyperplanes (lower dimension)

Lenstra's IP algorithm



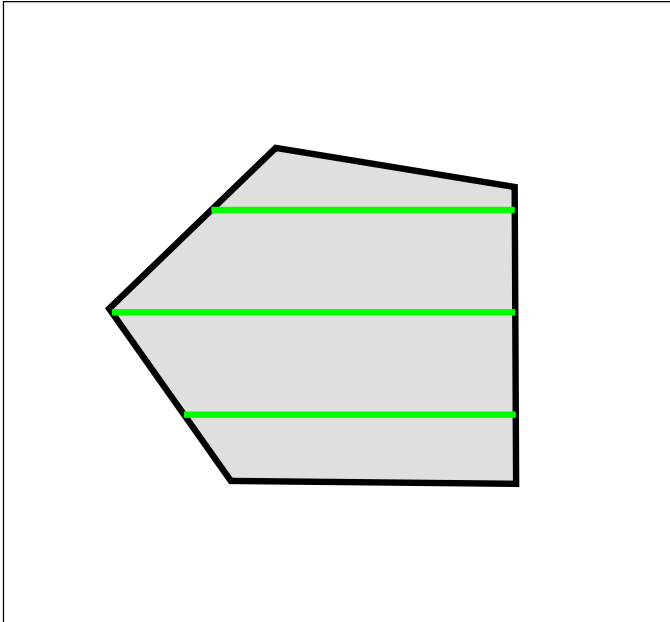
- Lenstra's algorithm is an algorithm for IP feasibility
- Computes width of polyhedron
- If width is too large, then return **feasible**
- Otherwise, **recursively** search for integer point on one of the constant number of hyperplanes (lower dimension)

Lenstra's IP algorithm



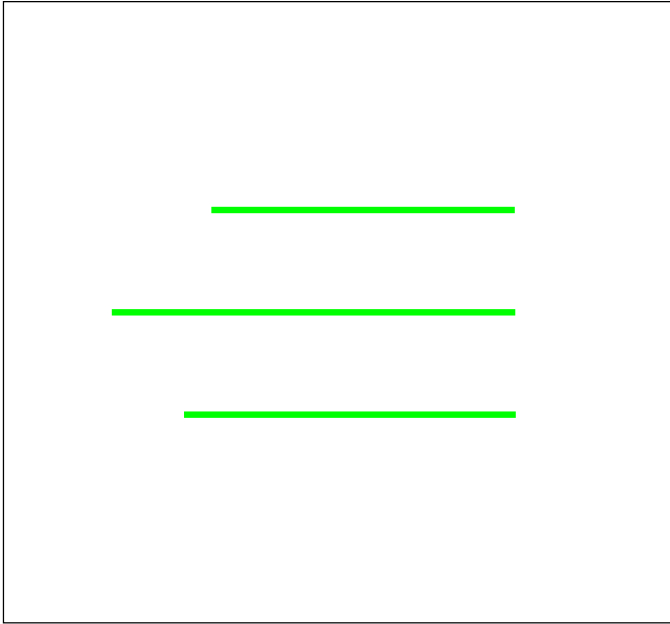
- Lenstra's algorithm is an algorithm for IP feasibility
- Computes width of polyhedron
- If width is too large, then return **feasible**
- Otherwise, **recursively** search for integer point on one of the constant number of hyperplanes (lower dimension)

Lenstra's IP algorithm



- Lenstra's algorithm is an algorithm for IP feasibility
- Computes width of polyhedron
- If width is too large, then return **feasible**
- Otherwise, **recursively** search for integer point on one of the constant number of hyperplanes (lower dimension)

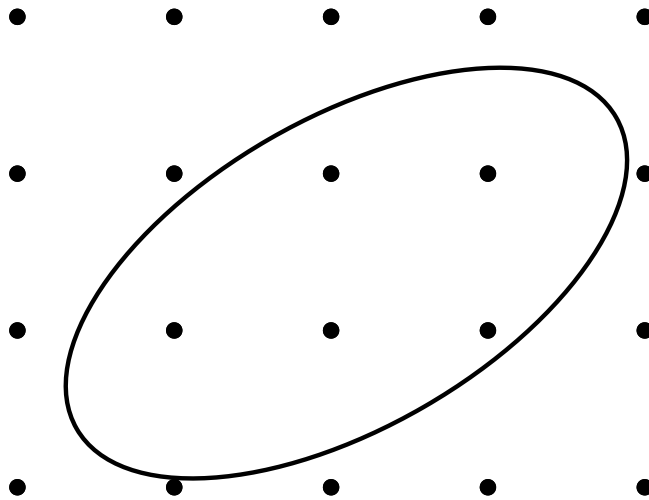
Lenstra's IP algorithm



- Lenstra's algorithm is an algorithm for IP feasibility
- Computes width of polyhedron
- If width is too large, then return **feasible**
- Otherwise, **recursively** search for integer point on one of the constant number of hyperplanes (lower dimension)

The feasibility problem for ellipsoids

- $A \in \mathbb{Q}^{n \times n}$ rational nonsingular matrix
- $a \in \mathbb{Q}^n$ rational vector
- $E(A, a) = \{x \in \mathbb{R}^n \mid \|A(x - a)\| \leq 1\}$ rational **ellipsoid** defined by A and a
- **Question:** $E(A, a) \cap \mathbb{Z}^n = \emptyset$?



A re-formulation

- $A \in \mathbb{Q}^{n \times n}$ rational nonsingular matrix
- $w \in \mathbb{Q}^n$ rational vector ($w = A a$)
- $\Lambda(A) = \{Ax \mid x \in \mathbb{Z}^n\}$
- **Question:** What is the point in $\Lambda(A)$ which is closest to w ?

Lattices

A **Lattice** is a set:

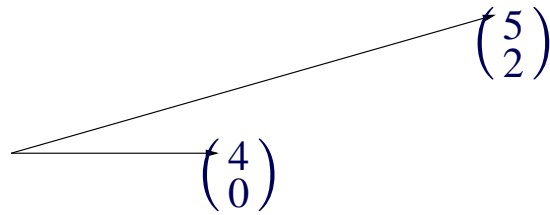
$$\Lambda(A) = \{Ax \mid x \in \mathbb{Z}^n\}$$

where $A \in \mathbb{Q}^{n \times n}$ is nonsingular rational matrix.

- A is **basis** of Λ .

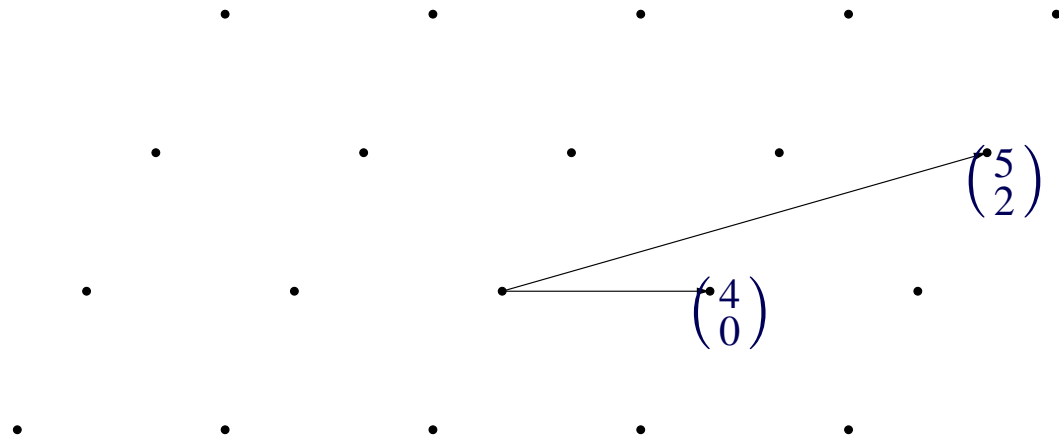
Lattices

$$A = \begin{pmatrix} 4 & 5 \\ 0 & 2 \end{pmatrix}$$



Lattices

$$A = \begin{pmatrix} 4 & 5 \\ 0 & 2 \end{pmatrix}$$



Exercise

- Let A be a lattice basis of Λ . Suppose that B originates from A by:
 - Swapping columns.
 - Subtracting **integer** multiples of a column from another column.

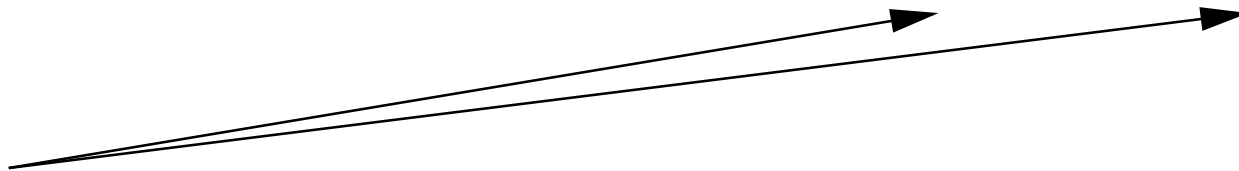
Show that B is also a basis of Λ .

- Show that $\begin{pmatrix} 4 & 5 \\ 0 & 2 \end{pmatrix}$ and $\begin{pmatrix} 4 & 1 \\ 0 & 2 \end{pmatrix}$ generate the same lattice. What is the shortest vector of this lattice?

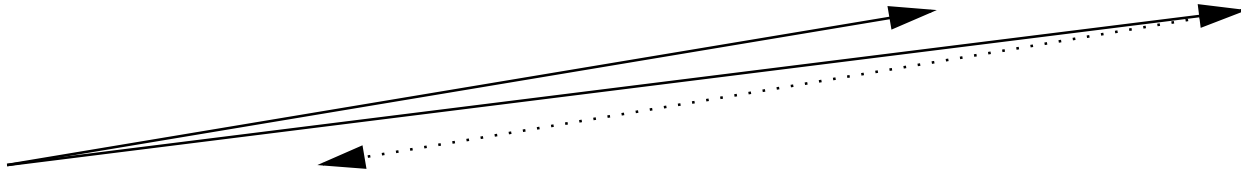
The central lattice problems

- Given a nonsingular matrix $A \in \mathbb{Q}^{n \times n}$ and a vector $w \in \mathbb{Q}^n$
- **Closest vector problem:** Determine $v \in \Lambda(A)$ with $\|v - w\|$ minimal **CV**
- **Shortest vector problem:** Determine $v \in \Lambda(A) - \{0\}$ with $\|v\|$ minimal **SV**

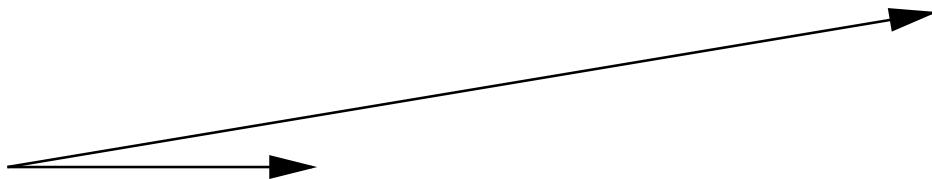
Quiz: What is the shortest vector?



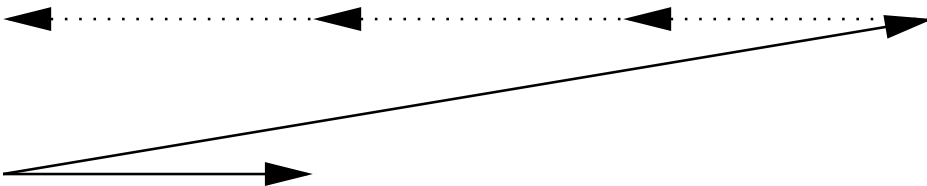
Quiz: What is the shortest vector?



Quiz: What is the shortest vector?



Quiz: What is the shortest vector?



Quiz: What is the shortest vector?



Exercise

- Let A be an orthogonal matrix, i.e., columns are orthogonal to each other. Show that the shortest vector of $\Lambda(A)$ w.r.t. ℓ_2 is the shortest vector of the basis A .

The lattice determinant

Let A and B be bases of Λ

- There exists integer matrix Q_1 such that $B = A Q_1$
- There exists integer matrix Q_2 such that $A = B Q_2$
- Thus $A = A Q_1 Q_2$, thus $Q_1 Q_2 = I_n$.
- $1 = \det(Q_1 Q_2) = \det(Q_1) \det(Q_2)$
- Q_1 and Q_2 integer matrices: $\det(Q_1), \det(Q_2) = \pm 1$
- $|\det(A)| = |\det(B)|$.

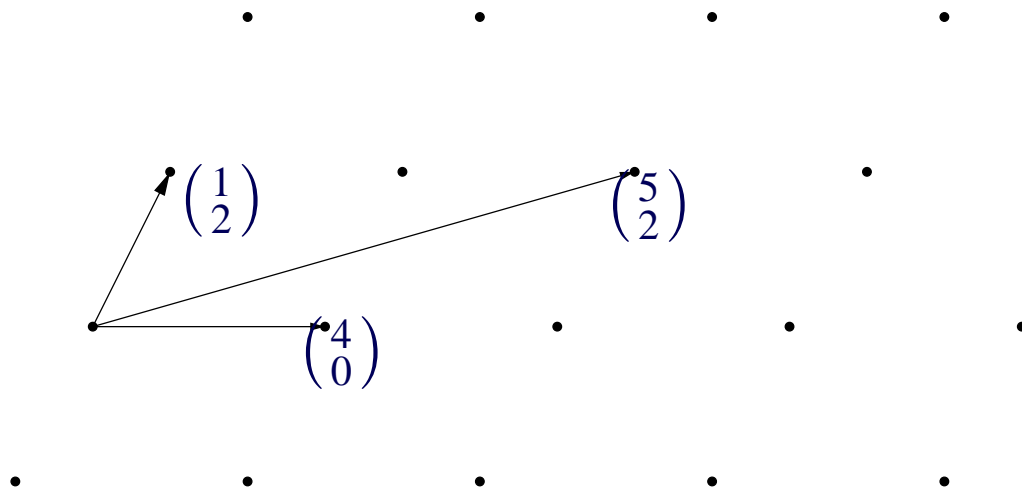
Lattice determinant:

$$\det(\Lambda) = |\det(A)|, \text{ where } A \text{ is basis of } \Lambda.$$

Example lattice determinant

- Lattice determinant is volume of parallelepiped of basis elements.

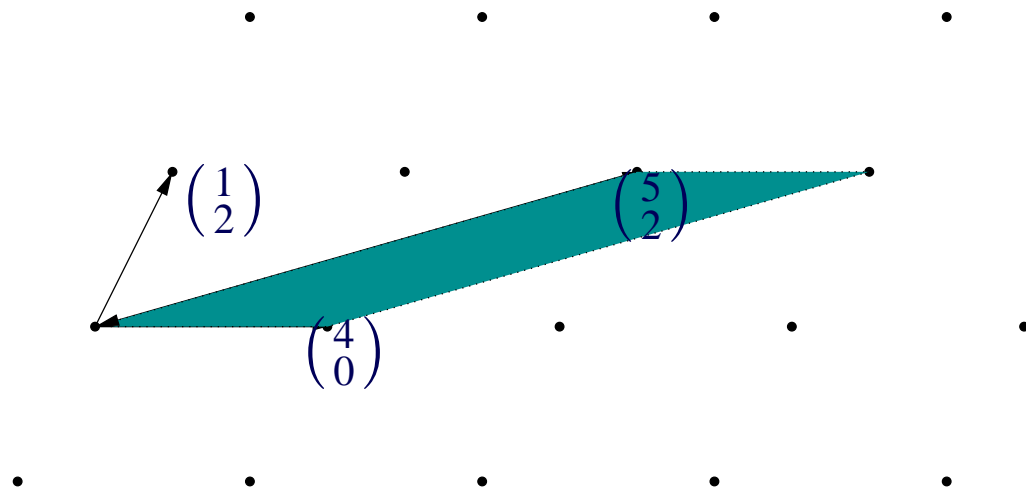
- The two bases below are $\begin{pmatrix} 4 & 5 \\ 0 & 2 \end{pmatrix}$ and $\begin{pmatrix} 4 & 1 \\ 0 & 2 \end{pmatrix}$.



Example lattice determinant

- Lattice determinant is volume of parallelepiped of basis elements.

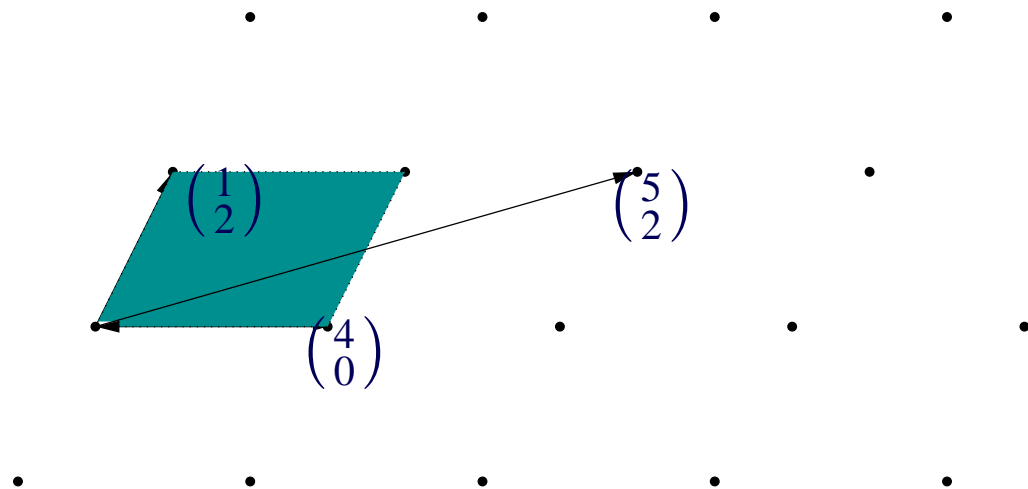
- The two bases below are $\begin{pmatrix} 4 & 5 \\ 0 & 2 \end{pmatrix}$ and $\begin{pmatrix} 4 & 1 \\ 0 & 2 \end{pmatrix}$.



Example lattice determinant

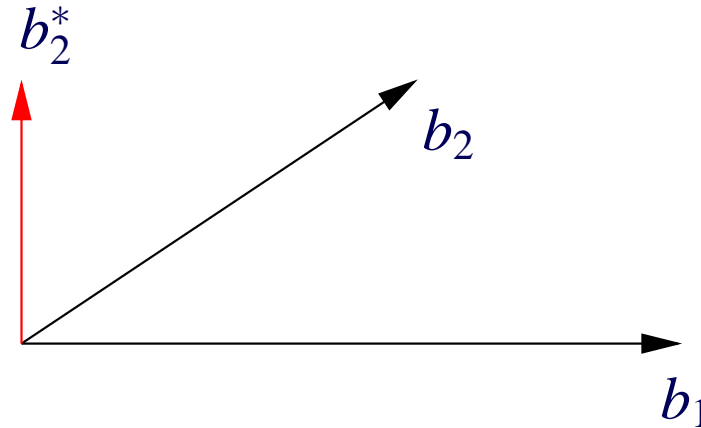
- Lattice determinant is volume of parallelepiped of basis elements.

- The two bases below are $\begin{pmatrix} 4 & 5 \\ 0 & 2 \end{pmatrix}$ and $\begin{pmatrix} 4 & 1 \\ 0 & 2 \end{pmatrix}$.



Gram-Schmidt orthogonalization

- b_1 and b_2 vectors.
- We search b_2^* orthogonal to b_1 s.t. (b_1, b_2^*) generate same vectorspace as (b_1, b_2)



- $b_2^* = b_2 - \mu b_1$
- $0 = \langle b_2^*, b_1 \rangle = \langle b_1, b_2 \rangle - \mu \langle b_1, b_1 \rangle$
- $\mu = \langle b_1, b_2 \rangle / \langle b_1, b_1 \rangle$

Gram-Schmidt orthogonalization

- Input: $b_1, \dots, b_n \in \mathbb{R}^n$
- Output: $b_1^*, \dots, b_n^* \in \mathbb{R}^n$ pairwise orthogonal and $\langle b_1^*, \dots, b_k^* \rangle = \langle b_1, \dots, b_k \rangle$ for all $k = 1, \dots, n$
- $b_1^* \leftarrow b_1$
- For $j = 2, \dots, k$
 - $b_j^* \leftarrow b_j - \sum_{i=1}^{j-1} \mu(i, j) b_i^*$, where $\mu(i, j)$ satisfies $\langle b_j - \mu(i, j) b_i^*, b_i^* \rangle = 0$

Gram-Schmidt orthogonalization

- Decomposes matrix $B \in \mathbb{Z}^{n \times n}$ into

$$B = B^* \begin{pmatrix} 1 & & \mu(i, j) \\ & \ddots & \\ 0 & & 1 \end{pmatrix},$$

where B^* is matrix with pairwise orthogonal columns.

Exercise

- Let $B = (b_1, \dots, b_{i-1}, b_i, b_{i+1}, b_{i+2}, \dots, b_n)$ and $\tilde{B} = (b_1, \dots, b_{i-2}, b_{i+1}, b_i, b_{i+2}, \dots, b_n)$ be two lattice bases. Notice that \tilde{B} originates from B via swapping the i -th and $i + 1$ -st column. Prove that B^* and \tilde{B}^* only differ in the i -th and $i + 1$ -st column.

Exercise

Let $B = B^* \begin{pmatrix} 1 & & \mu(i, j) \\ & \ddots & \\ 0 & & 1 \end{pmatrix}$ be the GSO of B

- Show that $\|b_i^*\| \leq \|b_i\|$ for $i = 1, \dots, n$
- Show that $|\det(B)| \leq \|b_1\| \cdots \|b_n\|$ where equality holds if and only if B is orthogonal (**Hadamard inequality**)

Orthogonality defect

Let $A \in \mathcal{Q}^{n \times n}$ be a nonsingular matrix. The number $\gamma \geq 1$ with $|\det(A)| \cdot \gamma = \|a_1\| \cdots \|a_n\|$ is the **orthogonality defect** of A .

Theorem. *A shortest vector of $\Lambda(A)$ is of the form*

$$\sum_{i=1}^n \lambda(i) a_i, \quad \text{where } \lambda(i) \in \mathbb{Z} \quad \text{and} \quad -\gamma \leq \lambda(i) \leq \gamma.$$

Proof

- Suppose that $|\lambda(n)| > \gamma$.
- Let $B = B^* \cdot R$ be the GSO of B
- Since $\|b_i\| \geq \|b_i^*\|$ and $\|b_1\| \cdots \|b_n\| = \gamma \cdot \|b_1^*\| \cdots \|b_n^*\|$ one has $\|b_n\| \leq \gamma \cdot \|b_n^*\|$
- $B\hat{\lambda} = B^* \cdot R\hat{\lambda} = u + \lambda(n) b_n^*$, where $\langle u, b_n^* \rangle = 0$
- Thus $\|B\hat{\lambda}\| \geq \lambda(n) \|b_n^*\| > \gamma \cdot \|b_n^*\| \geq \|b_n\|$ which is a contradiction to $B\hat{\lambda}$ being shortest vector

Small defect means SV is simple

Consequence:

Theorem. *Let $A \in \mathbb{Q}^{n \times n}$ be a rational lattice basis with orthogonality defect γ , then a shortest vector can be computed in time $O((2\gamma + 1)^n)$.*

Lattice basis reduction is a way to compute a basis B for $\Lambda(A)$ which has orthogonality defect $\leq d(n)$, where $d(n)$ is a number which depends only on the dimension.

Exercise

- Let A be an orthogonal matrix, i.e., columns are orthogonal to each other. Let $w \in \mathbb{Q}^n$. Suppose that $w = \sum_{i=1}^n \mu(i) a_i$. Show that the closest vector of $\Lambda(A)$ to w is the vector $\sum_{i=1}^n \lfloor \mu(i) \rfloor a_i$.

Small defect means CV is simple

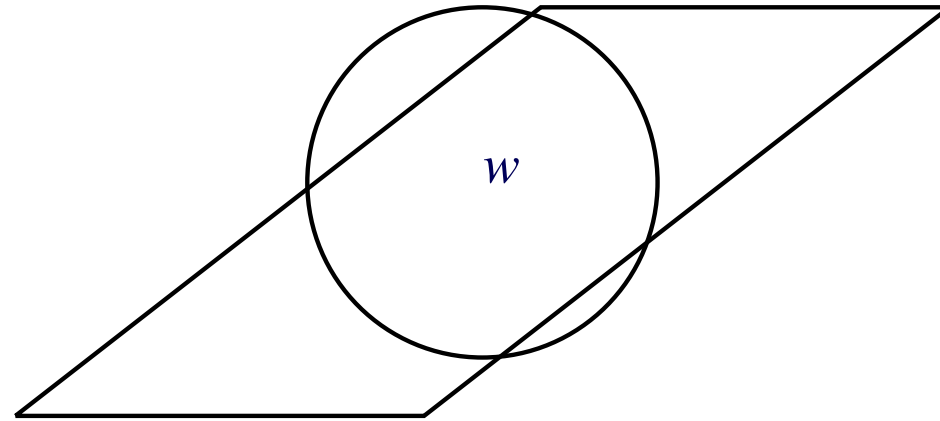
Theorem.

- *Let $A \in \mathbb{Q}^{n \times n}$ be a lattice basis with orthogonality defect γ .*
- *Suppose w.l.o.g. that last column a_n of A has largest norm*
- *Let $w \in \mathbb{Q}^n$ and let $B_{w,\varepsilon}$ be the ball of radius ε around w .*

If $B_{w,\varepsilon} \cap \Lambda(A) = \emptyset$, then $\|a_n^\| \geq \varepsilon / (\gamma \cdot n)$*

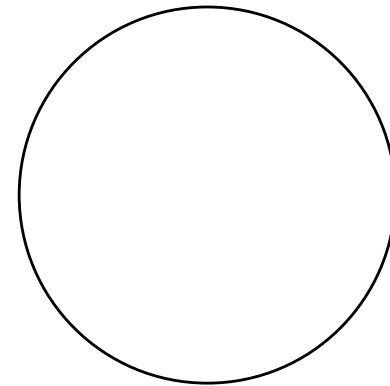
Proof

- $\|a_n^*\| \geq \|a_n\|/\gamma$
- **Suppose** $w = \sum_{i=1}^n \mu(i)a_i$
 $\mu(i) \in \mathbb{R}, 1 \leq i \leq n$
- $B_{w,\varepsilon} \cap \Lambda(A) = \emptyset \implies$
 $\sum_{i=1}^n (\mu(i) - \lfloor \mu(i) \rfloor)a_i \notin B_{0,\varepsilon}$.
- **Since** a_n **is largest basis**
vector $\implies \|a_n\| \geq \varepsilon/n$
- $\implies \|a_n^*\| \geq \varepsilon/(\gamma \cdot n)$



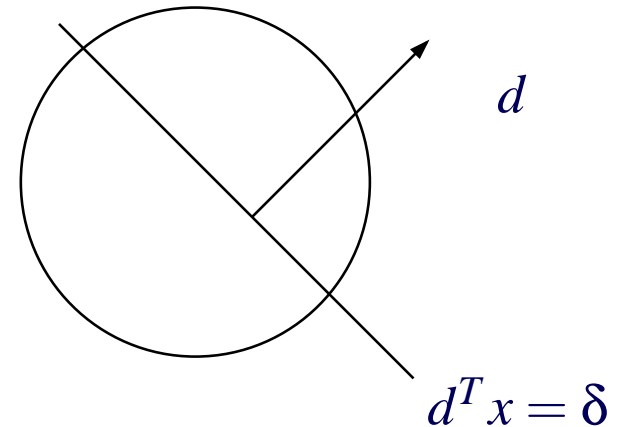
Searching the closest vector

- Let $d = a_n^* / \|a_n\|^2$
- $v \in \Lambda(A) \implies v = A^* \cdot R\lambda$,
where $\lambda \in \mathbb{Z}^n$
- $\implies d^T v = d^T A^* R\lambda =$
 $(0, \dots, 0, 1) R\lambda = \lambda(n) \in \mathbb{Z}$
- $\max\{d^T x \mid x \in B_{\varepsilon, w}\} - \min\{d^T x \mid$
 $x \in B_{\varepsilon, w}\} = 2\varepsilon \|d\| \leq 2\gamma n$
- **Recursively search** for lattice vector in $B_{\varepsilon, w} \cap (d^T x = \delta)$,
where $\delta \in \mathbb{Z}$ and $\max\{d^T x \mid x \in$
 $B_{\varepsilon, w}\} \geq \delta \geq \min\{d^T x \mid x \in B_{\varepsilon, w}\}$



Searching the closest vector

- Let $d = a_n^* / \|a_n\|^2$
- $v \in \Lambda(A) \implies v = A^* \cdot R\lambda$,
where $\lambda \in \mathbb{Z}^n$
- $\implies d^T v = d^T A^* R\lambda =$
 $(0, \dots, 0, 1) R\lambda = \lambda(n) \in \mathbb{Z}$
- $\max\{d^T x \mid x \in B_{\varepsilon, w}\} - \min\{d^T x \mid$
 $x \in B_{\varepsilon, w}\} = 2\varepsilon \|d\| \leq 2\gamma n$
- **Recursively search** for lattice vector in $B_{\varepsilon, w} \cap (d^T x = \delta)$,
where $\delta \in \mathbb{Z}$ and $\max\{d^T x \mid x \in$
 $B_{\varepsilon, w}\} \geq \delta \geq \min\{d^T x \mid x \in B_{\varepsilon, w}\}$



Basis reduction makes CV simple

Theorem (LLL Algorithm). *Let $A \in \mathbb{Q}^{n \times n}$ be a lattice basis. There exists an algorithm which runs in polynomial time (in binary input encoding) which computes a lattice basis $B \in \mathbb{Q}^{n \times n}$ with*

- $\Lambda(A) = \Lambda(B)$
- *orthogonality defect of B is $\leq 2^{n(n-1)/4}$*

If the dimension is fixed, the algorithm runs in linear time.

Solving CV

Theorem (Flatness theorem algorithmic version).

Let $A \in \mathbb{Q}^{n \times n}$, $w \in \mathbb{Q}^n$, $\varepsilon \in \mathbb{Q}_{>0}$.

There exists a polynomial algorithm which computes either

- $v \in \Lambda(A) \cap B_{\varepsilon, w}$ or
- $d \in \mathbb{Q}^n$ with $d^T v \in \mathbb{Z}$ for each $v \in \Lambda(A)$ and

$$\max\{d^T x \mid x \in B_{\varepsilon, w}\} - \min\{d^T x \mid x \in B_{\varepsilon, w}\} \leq 2n 2^{n(n-1)/4}.$$

If n is fixed the algorithm runs in linear time.

Proof

- $A \longrightarrow LLL \longrightarrow B$, let b_n be largest vector of B
- orthogonality defect $\gamma \leq 2^{n(n-1)/4}$
- $w = \sum_{i=1}^n \mu(i) b_i$
- $v = \sum_{i=1}^n \lfloor \mu(i) \rfloor b_i$
- $d = b_n^* / \|b_n^*\|^2$

Solving IP feasibility for ellipsoids

Theorem (Flatness theorem, Lenstra's algorithm).

Let $E(A, a)$ be a rational ellipsoid. There exists a polynomial algorithm which computes either

- *an integer point $x \in E(A, a) \cap \mathbb{Z}^n$*
- *or an integer vector $d \in \mathbb{Z}^n$ with*
$$\max\{d^T x \mid x \in E(A, a)\} - \min\{d^T x \mid x \in E(A, a)\} \leq 2n 2^{n(n-1)/4}$$

If the dimension n is fixed, the algorithm runs in linear time.

proof

- Find vector in $\Lambda(A) \cap B_{1, Aw}$ or
- Find vector f with $f^T A \in \mathbb{Z}^n$ and

$$\max\{f^T x \mid x \in B_{1,w}\} - \min\{f^T x \mid x \in B_{1,w}\} \leq 2n2^{n(n-1)/4}.$$

- $f^T x = f^T A A^{-1} x$
- $E(A, a) = A^{-1} B_{1,w}$
- $d^T = f^T A$

Solving IP feasibility for ellipsoids

Theorem. *If the dimension n is fixed, there exists a linear-time algorithm to solve the IP-feasibility problem for ellipsoids.*

- Algorithm above either determines integer point or determines $d \in \mathbb{Z}^n$ with
$$\max\{d^T x \mid x \in E(A, a)\} - \min\{d^T x \mid x \in E(A, a)\} \leq 2n 2^{n(n-1)/4}$$
- We can assume $\gcd(d) = 1$
- Compute unimodular matrix U with $d^T U = (1, 0, \dots, 0)$
- $E(A, a) \cap (d^T x = \delta)$ contains integer point if and only if $E(AU^{-1}, a) \cap x(1) = \delta$ contains integer point. (ellipsoid in lower dimension)
- **Exercise:** Give a closed formula for the ellipsoid above in $n - 1$ variables.

The flatness theorem for convex bodies

Max. volume ellipsoids

- Each convex body $K \subseteq \mathbb{R}^n$ has a unique **max. volume ellipsoid** $E(A, b)$ contained in K .
- $E(A/n, a) \supseteq K \supseteq E(A, a)$

Theorem (Flatness theorem convex bodies). *Let $K \subseteq \mathbb{R}^n$ be a convex body. If $K \cap \mathbb{Z}^n \neq \emptyset$ then there exists a $d \in \mathbb{Z}^n - \{0\}$ such that*

$$\max\{d^T x \mid x \in K\} - \min\{d^T x \mid x \in K\} \leq 2n^2 2^{n(n-1)/4}.$$

Better flatness constants

- For Ellipsoids: $O(n)$ (Banaszczyk 1996)
- Simplices: $O(n \log n)$ (Banaszczyk & Litvak 1999)

The LLL algorithm

A lower bound on SV_2

Theorem. *Let B be a lattice basis and let $B^* = (b_1^*, \dots, b_n^*)$ be its Gram-Schmidt orthogonalization, then $SV_2(\Lambda(B)) \geq \min_{j=1, \dots, n} \|b_j^*\|_2$.*

Proof of theorem

- $0 \neq v \in \Lambda$, then $v = \sum_{j=1, \dots, k} \lambda(j) b_j$, where $k \leq n$, $\lambda(j) \in \mathbb{Z}$, $j = 1, \dots, k$ and $x_k \neq 0$.
- Using GSO:

$$\begin{aligned} v &= \sum_{j=1, \dots, k} \left(\lambda(j) \left(b_j^* + \sum_{i=1}^{j-1} \mu_{ij} b_i^* \right) \right) \\ &= \lambda(k) b_k^* + \sum_{i=1}^{k-1} x(i) b_i^*, \text{ for some } x(i) \in \mathbb{R}. \end{aligned}$$

$$\|v\| = |\lambda(k)| \|b_k^*\| + \sum_{i=1}^{k-1} |x(i)| \|b_i^*\| \geq \|b_k^*\|.$$

Summary of insight progress

- If lattice basis is orthogonal, shortest vector is easy.
- The Gram-Schmidt orthogonalization of lattice basis provides lower bound on shortest vector.
- First vector of GSO is first vector of basis.
- Typically vectors in GSO B^* of B are decreasing rapidly, thus spoiling the lower bound.

Natural conclusion

- Given a basis, turn it into something which resembles as much as possible to its GSO.
- Try to assure that the vectors in GSO do not decrease fast, so that first vector is about the size of the minimum in GSO.

The LLL Algorithm

- Normalize: Subtract integer multiples of columns from another column so that $|\mu_{ij}| \leq 1/2$ for every $1 \leq i < j \leq n$ in GSO decomposition

$$B = B^* \begin{pmatrix} 1 & & \mu \\ & \ddots & \\ 0 & & 1 \end{pmatrix}.$$

- Swap (fight the decrease of the $\|b_j^*\|$): If there exists a j such that

$$\|b_{j+1}^* + \mu_{j,j+1} b_j^*\|^2 < 3/4 \|b_j^*\|^2,$$

swap b_j and b_{j+1} . Goto Normalize

Swap: Explanation

- The vector $b_{j+1}^* + \mu_{j,j+1} b_j^*$ is the new j -th vector of B^* after the swap because
 - $b_{j+1}^* = b_{j+1} - \sum_{i=1}^j \mu_{i,j+1} b_i^*$.
 - The vector $b_{j+1}^* + \mu_{j,j+1} b_j^*$ is projection of b_{j+1} into orthogonal complement of b_1, \dots, b_{j-1} .
 - The vector $b_{j+1}^* + \mu_{j,j+1} b_j^*$ is new j -th column of B^* after the swap.
 - The j -th column decreases by $3/4$
 - The only possible side effect is an increase of the $j+1$ -st column. The rest of the GSO remains unchanged.

A potential function

- $\phi(B) = \|b_1^*\|^{2n} \|b_2^*\|^{2(n-1)} \|b_3^*\|^{2(n-2)} \dots \|b_n^*\|^2$
- Define $B_j = (b_1, \dots, b_j)$
- $\|b_1^*\| \dots \|b_j^*\|$ is j -dimensional volume of parallelepiped of b_1, \dots, b_j
- $\det(B_j^T B_j) = \|b_1^*\|^2 \dots \|b_j^*\|^2 \in \mathbb{Z}$
- $\phi(B) = \prod_{j=1}^n \det(B_j^T B_j) \in \mathbb{Z}$
- A swap of b_j and b_{j+1} of LLL does not change volume for $i = 1, \dots, j-1, j+1, \dots, n$, and decreases $\det(B_j^T B_j)$ by a factor of $3/4$.
- Since $\phi(B)$ is an integer (all bases remain integral during LLL) the algorithm terminates in a polynomial number of steps.

Termination of the LLL

We just proved:

Theorem. *Given an integer lattice basis B , the LLL algorithm performs a polynomial number of steps.*

What remains to be done:

- Need to argue that the binary encoding length of numbers involved remains polynomial.

The first vector is short

Let B be a basis returned by LLL:

-

$$\begin{aligned} 3/4 \|b_j^*\|^2 &\leq \|b_{j+1} + \mu_{j,j+1} b_j^*\|^2 \\ &\leq \|b_{j+1}\|^2 + 1/4 \|b_j^*\|^2 \end{aligned}$$

- Thus $\|b_j^*\|^2 \leq 2 \|b_{j+1}^*\|^2$ (we successfully fought the rapid decrease!)
- $\|b_1\| = \|b_1^*\| \leq 2^{(n-1)/2} \min\{\|b_i^*\| \mid i = 1, \dots, n\}$
- Thus $\|b_1\| \leq 2^{(n-1)/2} \text{SV}(\Lambda(B))$
- Thus SV can be approximated within a factor of $2^{(n-1)/2}$ in polynomial time

The binary encoding length

What follows is a sketch of polynomiality.

- After normalization:

$$\|b_j\|^2 = \sum_{i=1}^j \mu_{ij}^2 \|b_i^*\|^2 \leq \sum_{i=1}^j \|b_i^*\|^2 \leq n \det(\Lambda)$$

- b_j are integral vectors, together with fact above, their encoding length is polynomial in input. (Remember Hadamard bound)!
- GSO is polynomial operation.
- The normalization is polynomial, because it operates on upper-right matrix in GSO decomposition.

The complexity of the LLL

Conclusion:

- The LLL algorithm is polynomial in the bit model of computation.
- If the dimension is fixed, it runs in linear time in binary encoding length (as the Euclidean algorithm)

Orthogonality defect; Exercise

- Show that LLL basis B satisfies

$$\prod_{i=1}^n \|b_i\| \leq 2^{\binom{n}{2}/2} |\det(B)|.$$

Basis reduction, historical notes

- Lattice basis reduction has its origin in the work of Lagrange on binary quadratic forms.
- With a technical, but algorithmic proof, Gauß [Gau01] showed that a 3-dimensional lattice Λ has a nonzero vector of length $(4/3)^{1/2} \det(\Lambda)^{1/3}$
- Hermite [Her50] generalized this result by showing that each n -dimensional lattice Λ has a nonzero vector v , such that $\|v\|_2 \leq (4/3)^{n-1/4} \det(\Lambda)^{1/n}$.

Basis reduction, historical notes

- The LLL algorithm is by Lenstra, Lenstra and Lovász [LLL82].
- A non-algorithmic and beautiful proof of these facts was given by Minkowski [Min68], who opened the stage for a new discipline of mathematics, the geometry of numbers.

Exercise

- Show that, given a lattice basis B , one can in polynomial time compute a nonzero vector $v \in \Lambda(B) - \{0\}$, such that $\|v\| \leq 2^{n-1/4} \sqrt[n]{|\det(B)|}$.

Computing the width of a simplex

The width of a simplex

- Since translation leaves flatness invariant 0 is a vertex
- $\Sigma = \text{conv}\{0, v_1, \dots, v_n\}$ **Simplex.**
- A matrix with rows v_1^T, \dots, v_n^T
- width of Σ along c :

$$\|A c\|_\infty \leq w_c(\Sigma) \leq 2 \|A c\|_\infty$$

Minimal width along integer $c \in \mathbb{Z}^n - \{0\} \approx$ length of shortest vector of $\Lambda(A) = \{A c \mid c \in \mathbb{Z}^n\}$.

Exercise

- Let Σ be a simplex in fixed dimension. Show that one can determine an integer direction $d \in \mathbb{Z}^n - \{0\}$ with

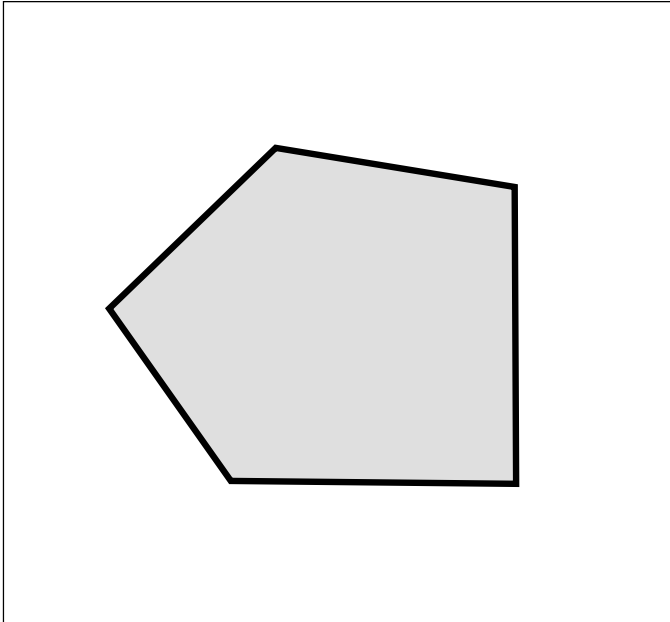
$$\max\{d^T x \mid x \in \Sigma\} - \min\{d^T x \mid x \in \Sigma\}$$

in **linear time**.

- Hint: Given an LLL-reduced basis in fixed dimension and a constant α , one can enumerate all vectors whose length is at most α times the shortest vector length in constant time.

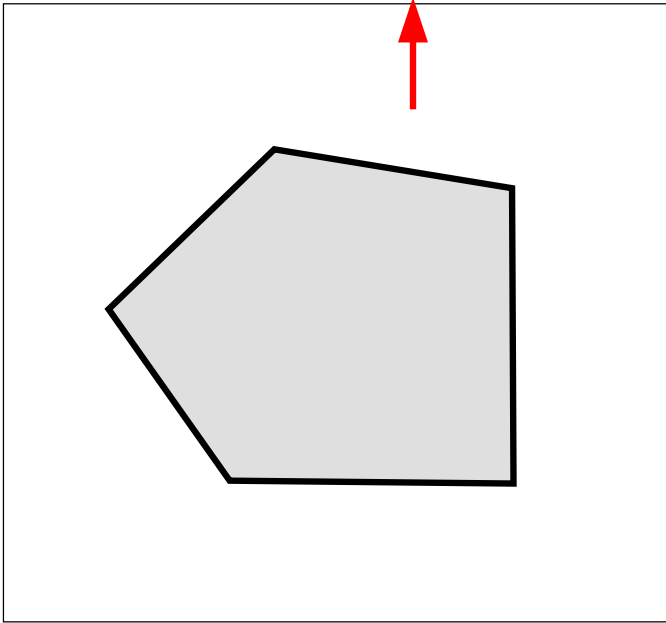
Solving the optimization problem efficiently

Lenstra's IP algorithm



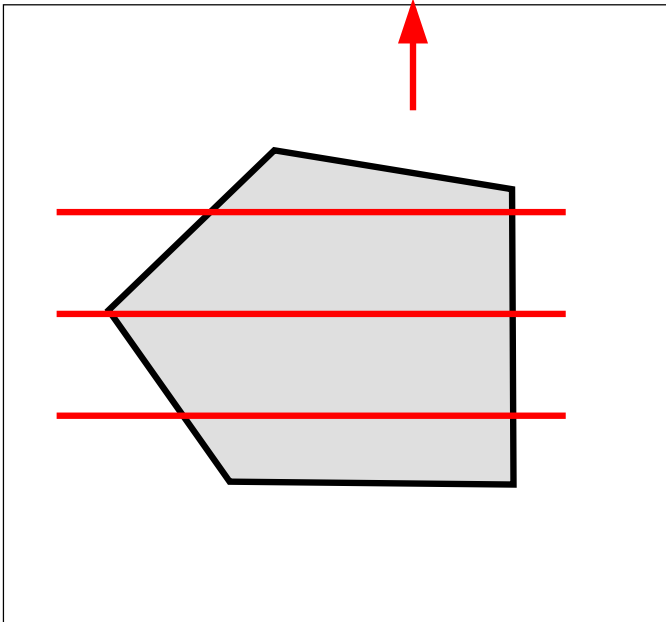
- Lenstra's algorithm is an algorithm for IP feasibility
- Computes width of polyhedron
- If width is too large, then return **feasible**
- Otherwise, **recursively** search for integer point on one of the constant number of hyperplanes (lower dimension)

Lenstra's IP algorithm



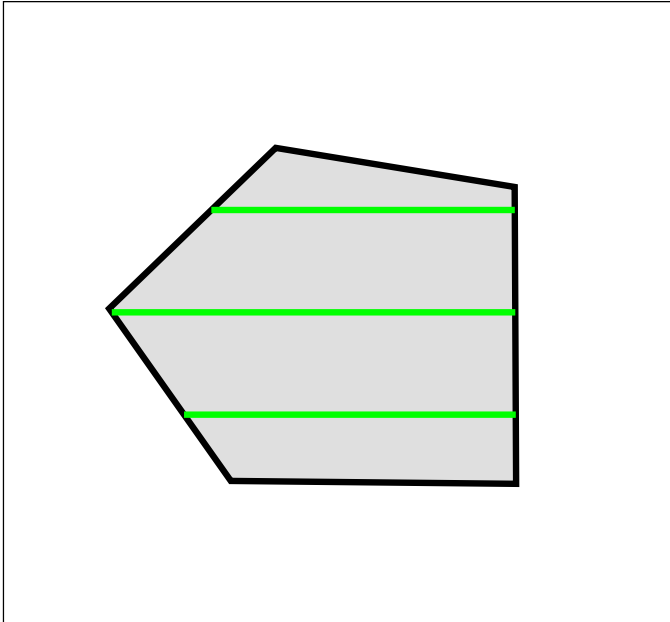
- Lenstra's algorithm is an algorithm for IP feasibility
- Computes width of polyhedron
- If width is too large, then return **feasible**
- Otherwise, **recursively** search for integer point on one of the constant number of hyperplanes (lower dimension)

Lenstra's IP algorithm



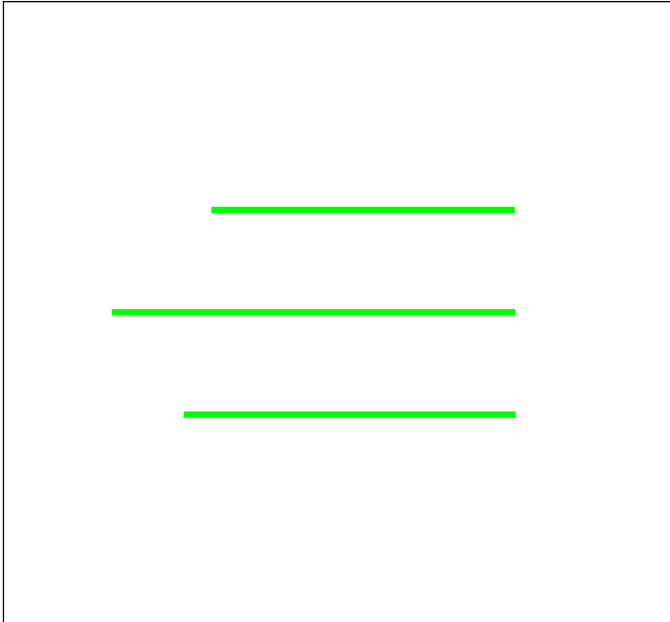
- Lenstra's algorithm is an algorithm for IP feasibility
- Computes width of polyhedron
- If width is too large, then return **feasible**
- Otherwise, **recursively** search for integer point on one of the constant number of hyperplanes (lower dimension)

Lenstra's IP algorithm



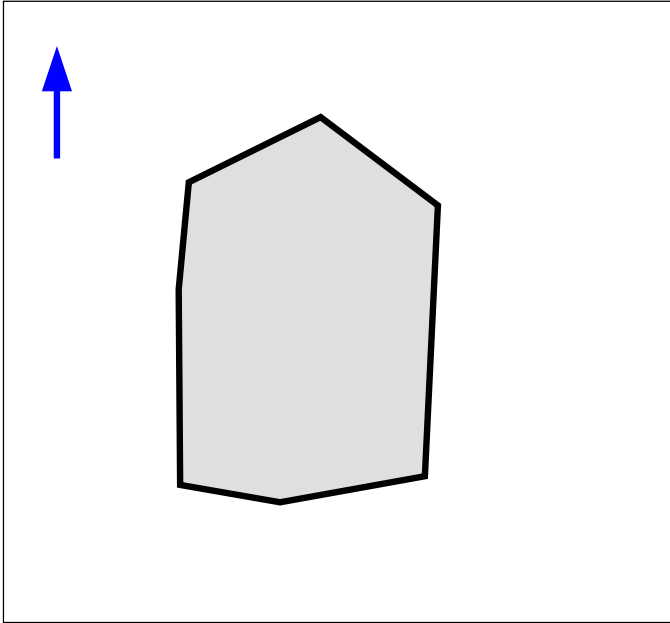
- Lenstra's algorithm is an algorithm for IP feasibility
- Computes width of polyhedron
- If width is too large, then return **feasible**
- Otherwise, **recursively** search for integer point on one of the constant number of hyperplanes (lower dimension)

Lenstra's IP algorithm



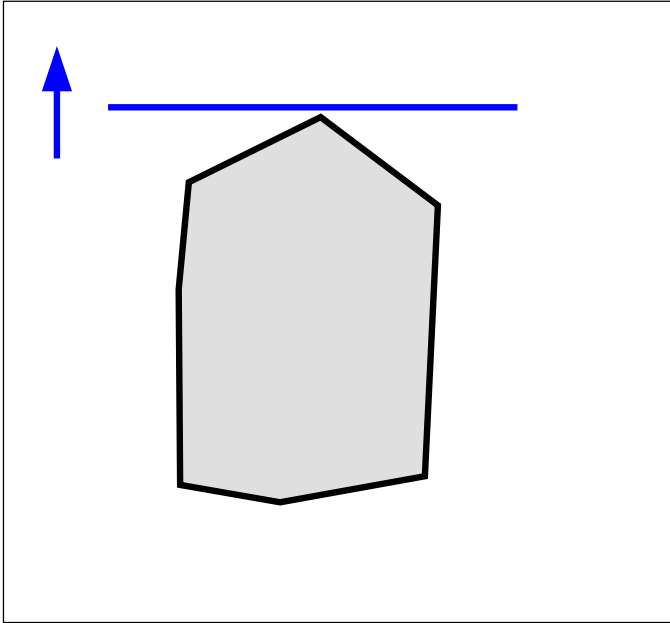
- Lenstra's algorithm is an algorithm for IP feasibility
- Computes width of polyhedron
- If width is too large, then return **feasible**
- Otherwise, **recursively** search for integer point on one of the constant number of hyperplanes (lower dimension)

Sliding objective



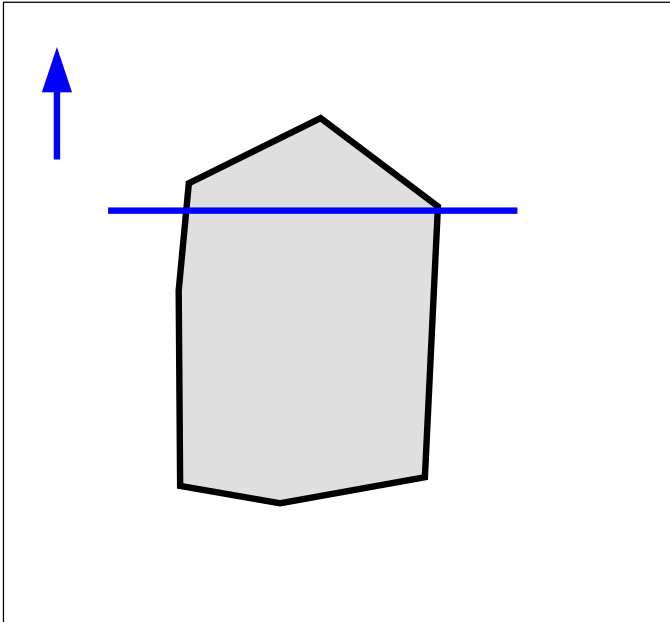
- Let objective function slide into polyhedron
- Until truncation is not flat anymore
- **Optimum** lies on one of a constant number of hyperplanes
- Continue search for optimum in the hyperplanes

Sliding objective



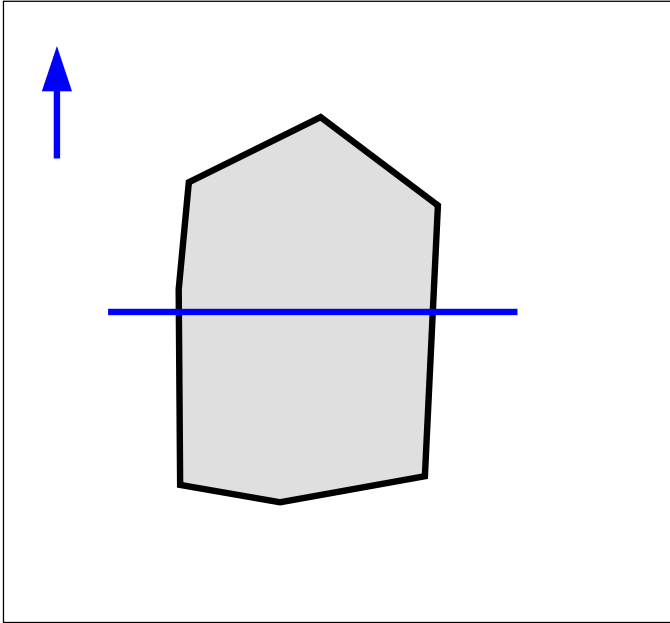
- Let objective function slide into polyhedron
- Until truncation is not flat anymore
- **Optimum** lies on one of a constant number of hyperplanes
- Continue search for optimum in the hyperplanes

Sliding objective



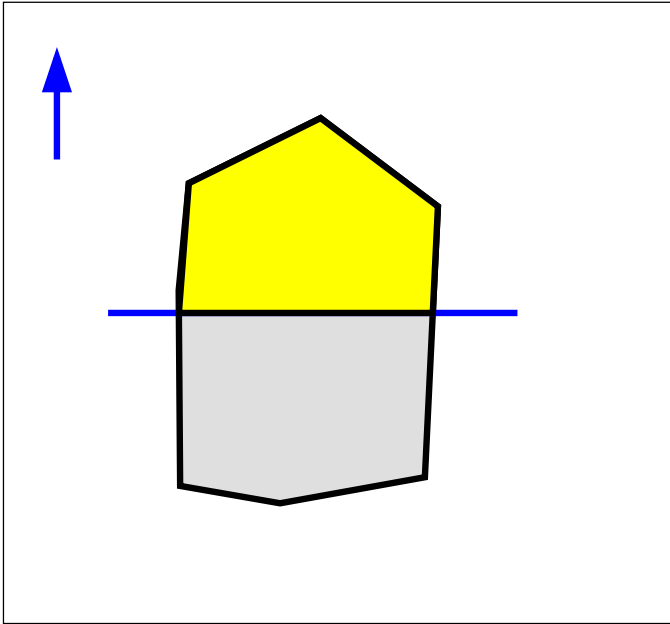
- Let objective function slide into polyhedron
- Until truncation is not flat anymore
- **Optimum** lies on one of a constant number of hyperplanes
- Continue search for optimum in the hyperplanes

Sliding objective



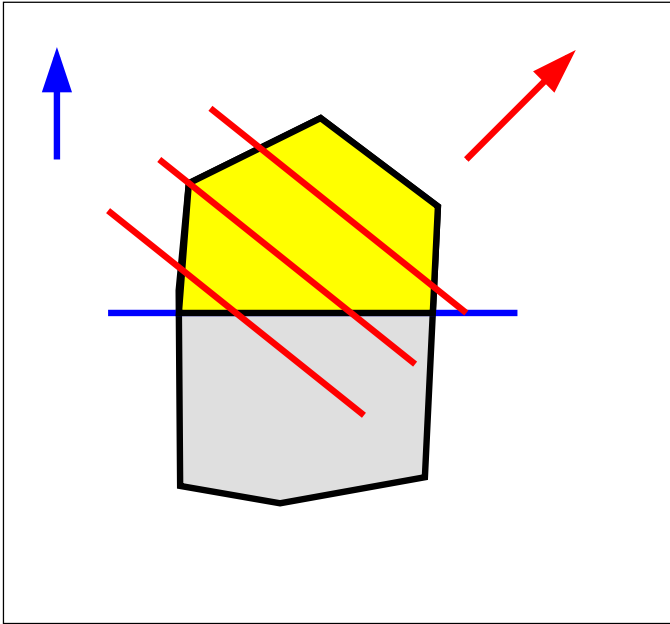
- Let objective function slide into polyhedron
- Until truncation is not flat anymore
- **Optimum** lies on one of a constant number of hyperplanes
- Continue search for optimum in the hyperplanes

Sliding objective



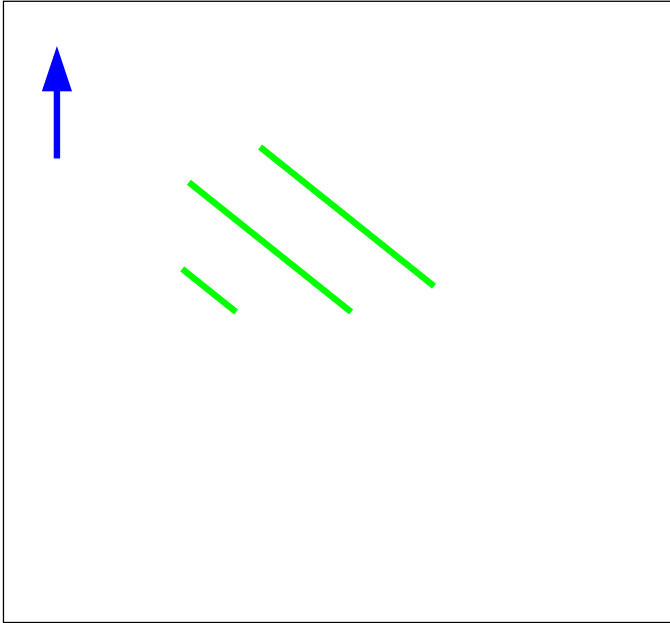
- Let objective function slide into polyhedron
- Until truncation is not flat anymore
- **Optimum** lies on one of a constant number of hyperplanes
- Continue search for optimum in the hyperplanes

Sliding objective



- Let objective function slide into polyhedron
- Until truncation is not flat anymore
- **Optimum** lies on one of a constant number of hyperplanes
- Continue search for optimum in the hyperplanes

Sliding objective



- Let objective function slide into polyhedron
- Until truncation is not flat anymore
- **Optimum** lies on one of a constant number of hyperplanes
- Continue search for optimum in the hyperplanes

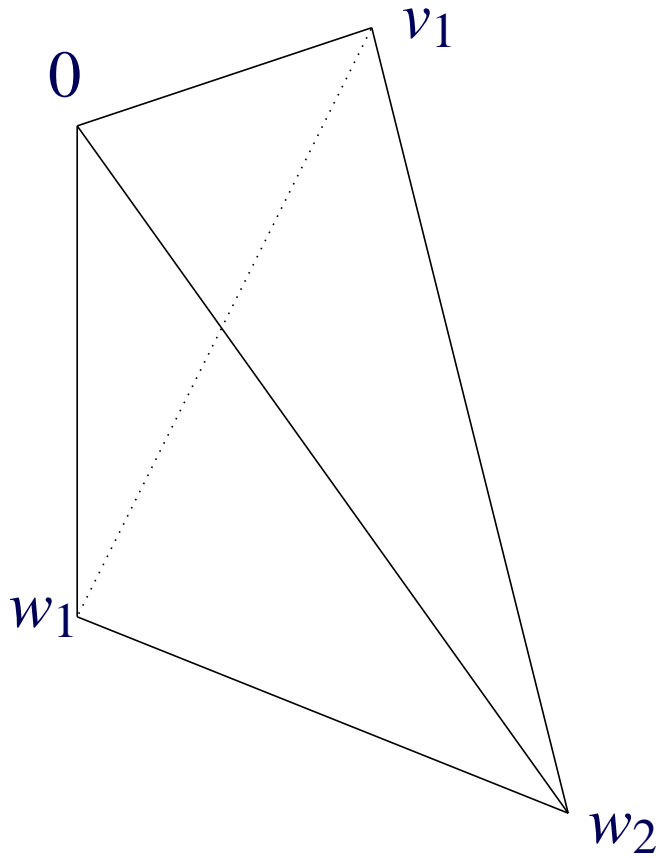
Problem: Geometry of truncation changes too much in the sliding process

Key idea: Restrict to two-layer simplices

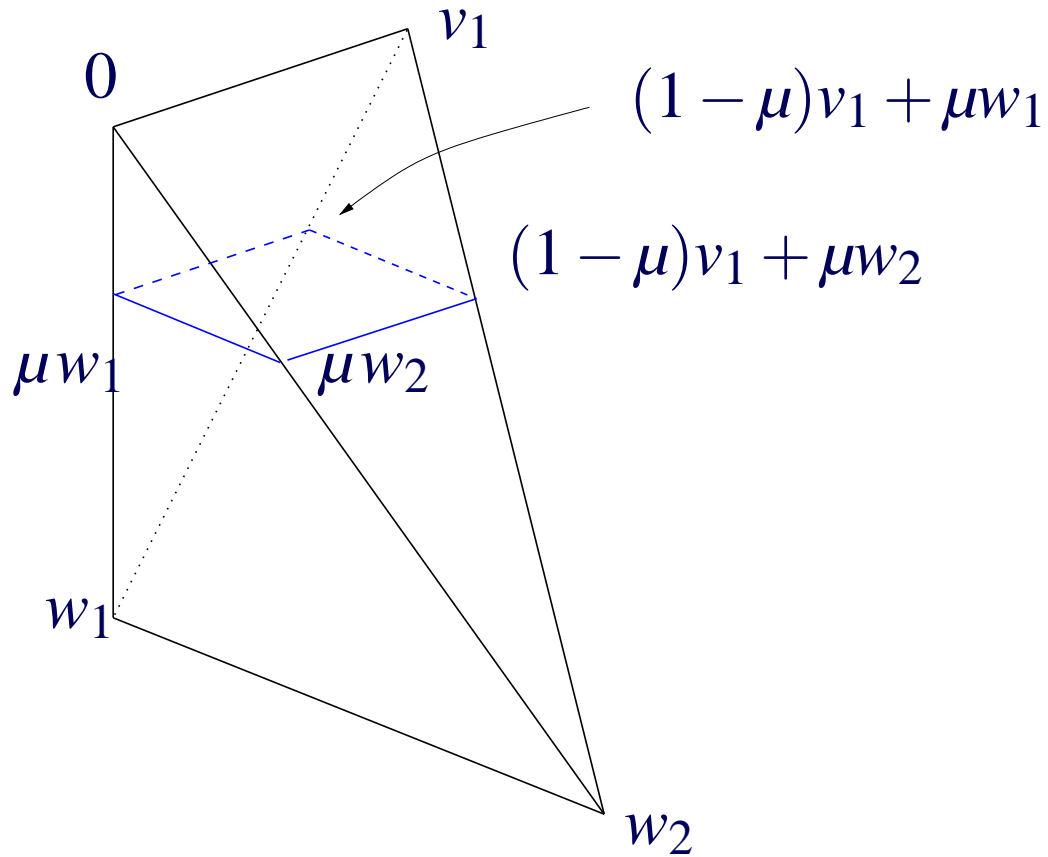
- Σ is two-layer simplex if vertices partition into two sets V and W such that

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

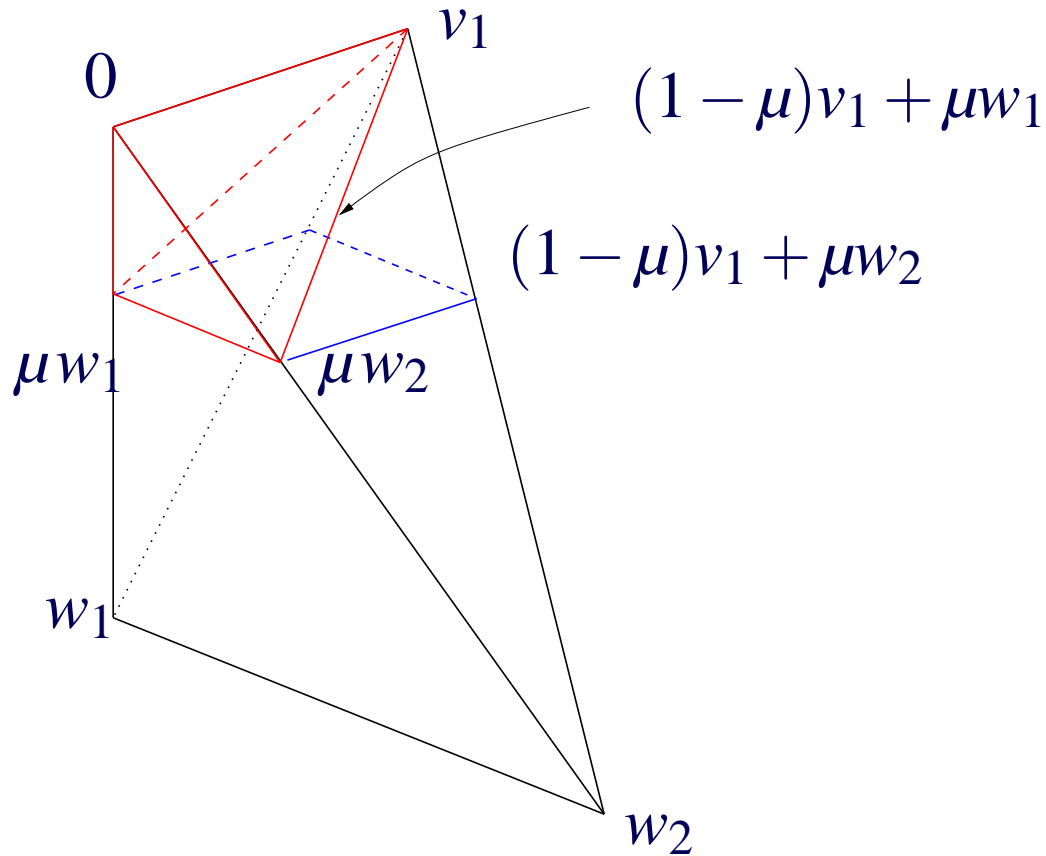
Example in 3D



Example in 3D



Example in 3D



Width of truncation is approximately width of simplex spanned by V and μW .

Width and shortest vector

Width of $\Sigma_{V,\mu W}$ is “about” the length of shortest vector in $\Lambda(A_{\mu,k})$, where

- A is matrix with rows $w_1^T, \dots, w_k^T, v_1^T, \dots, v_{n-k}^T$
- $A_{\mu,k}$ results from A by scaling first k rows with μ

Parametric shortest vector

The following problem has to be solved:

PARAMETRIC SHORTEST VECTOR:

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

Theorem (E. 2003). *The parametric shortest vector problem can be solved in linear time in fixed dimension.*

Algorithm for PSV

```
 $p \leftarrow 2^{\lceil \log U \rceil}$   
 $B \leftarrow A_{p,k}$   
repeat  
  if  $p = 1$   
    return  $SV(\Lambda) > U$   
   $B \leftarrow B_{1/2,k}$   
   $p \leftarrow p/2$   
   $B \leftarrow \text{LLL}(B)$   
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 strictly decreases
- $B_1 \rightarrow \text{LLL} \rightarrow B_2$: $\log \phi(B_1) - \log \phi(B_2)$ iterations
- $\phi(A_{U,k}) \leq \phi(A_{U,n}) \leq U^{2n^2} (\|a_1\| \cdots \|a_n\|)^{2n}$
- In fixed dimension $O(\text{size}(U) + \text{size}(A))$ iterations, **linear**

Complexity of IP any fixed dimension

Using Clarkson's algorithm for LP-type problems one then obtains:

Theorem (E. 2003). *An integer program with m constraints, each involving coefficients of size at most s can be solved in expected time $O(m + (\log m)s)$.*

What have we learned in these lectures ?

- We know why and how to reduce a basis
- We know how to compute shortest and closest vectors in fixed dimension in linear time
- We have learned about Prune&Search and about Clarkson's random sampling algorithm
- We have seen that IP in fixed dimension can be solved with an **almost optimal algorithm**

Effient algorithms for the plane

History

m: Number of constraints

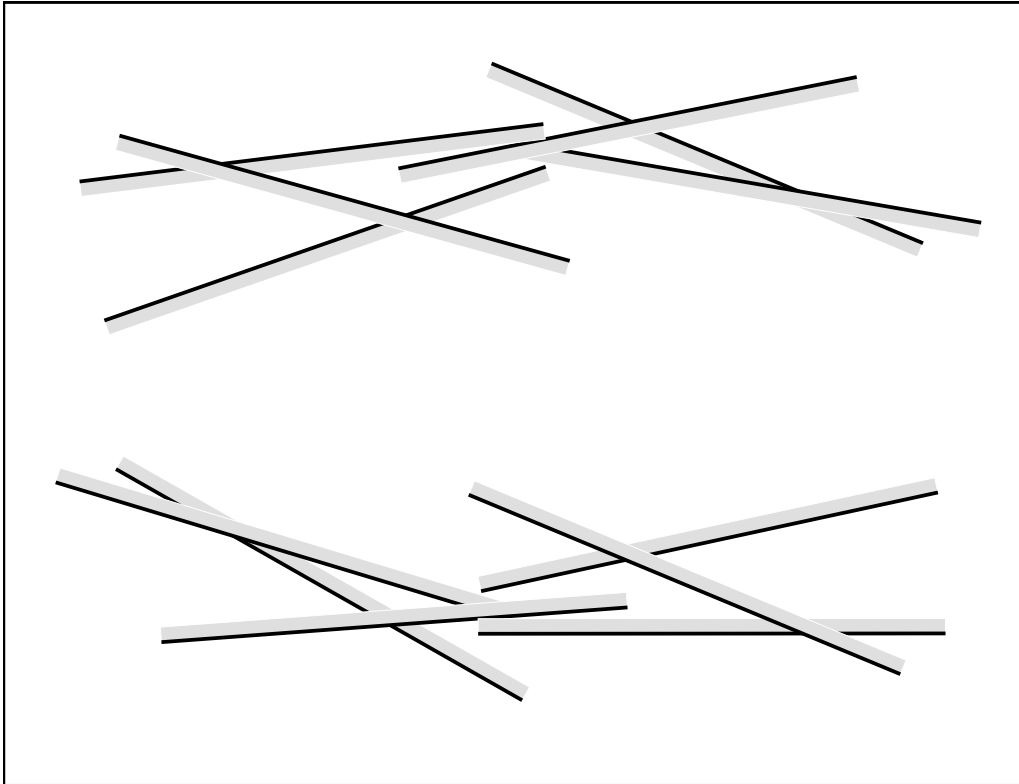
s: largest binary encoding length of coefficient

Method	Complexity
Kannan 1980, Scharf 1981	polynomial
Lenstra 1983	$O(ms + s^2)$
Feit 1984	$O(m \log m + ms)$
Zamanskij and Cherkasskij 1984	$O(m \log m + ms)$
Kanamaru, Nishizeki and Asano 1994	$O(m \log m + s)$
E. and Rote 2000	$O(m + (\log m)s)$
E. 2003	$O(m + (\log m)s)$
E. & Laue	$O(m + s)$
Feasibility test + Euclidean algorithm	$O(m + s)$

any fixed dimension

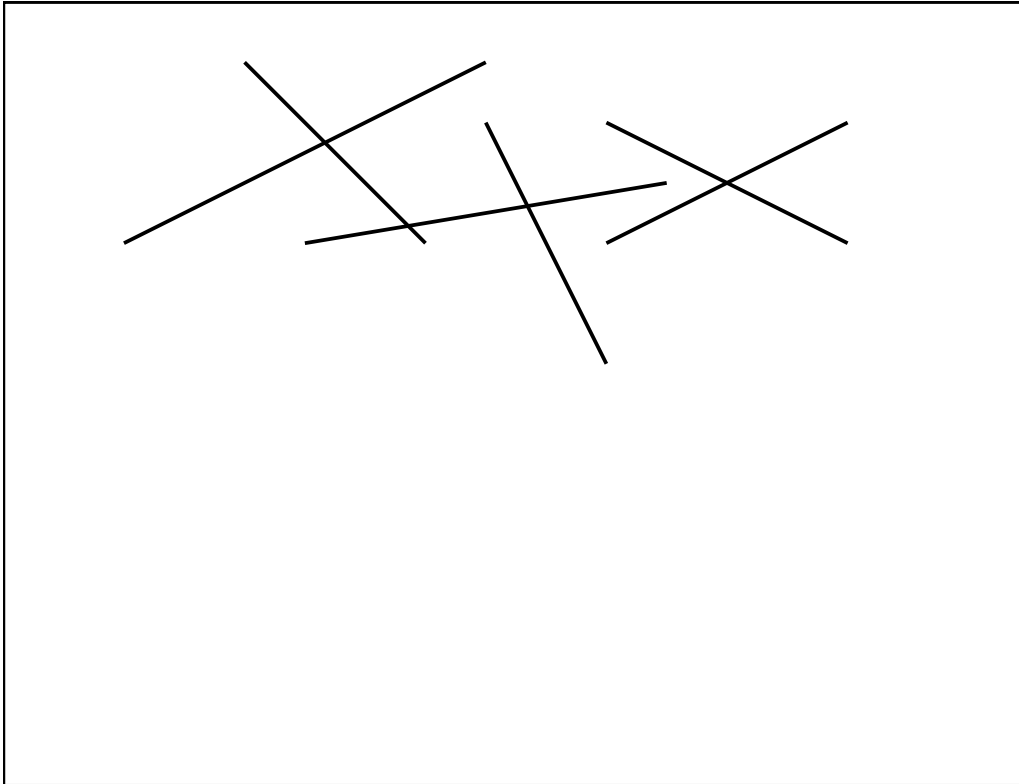
Prune & Search: Dealing with the combinatorics

Megiddo's Algorithm for LP in the plane



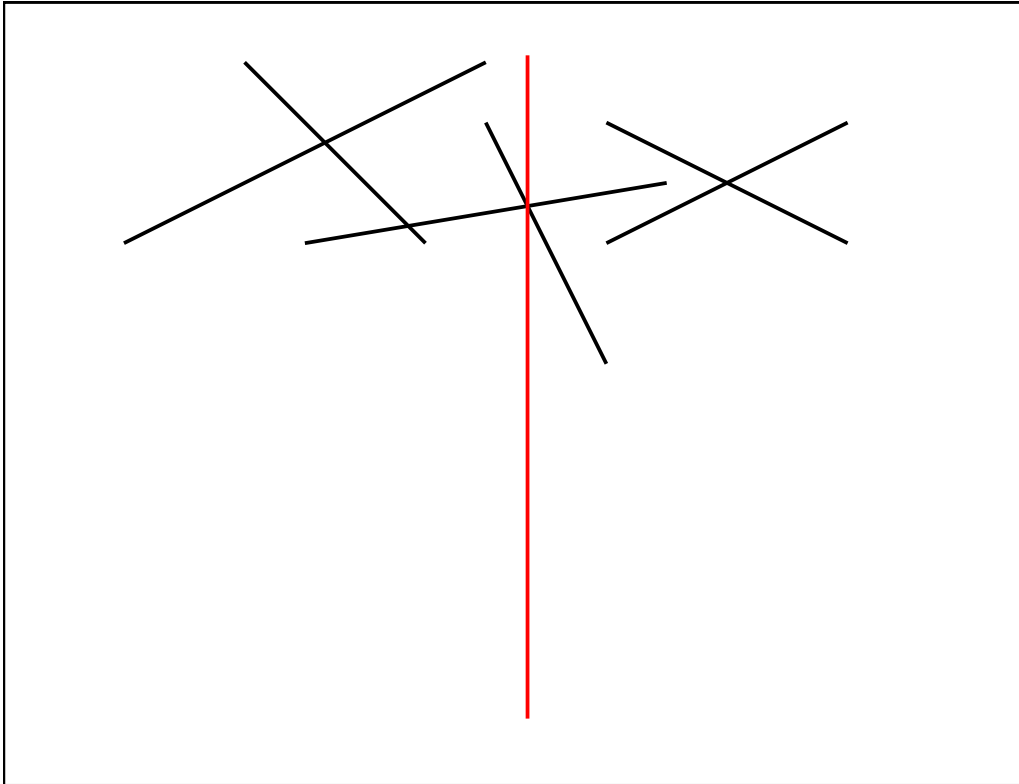
- Partition constraints into “down” and “up” constraints

Megiddo's Algorithm for LP in the plane



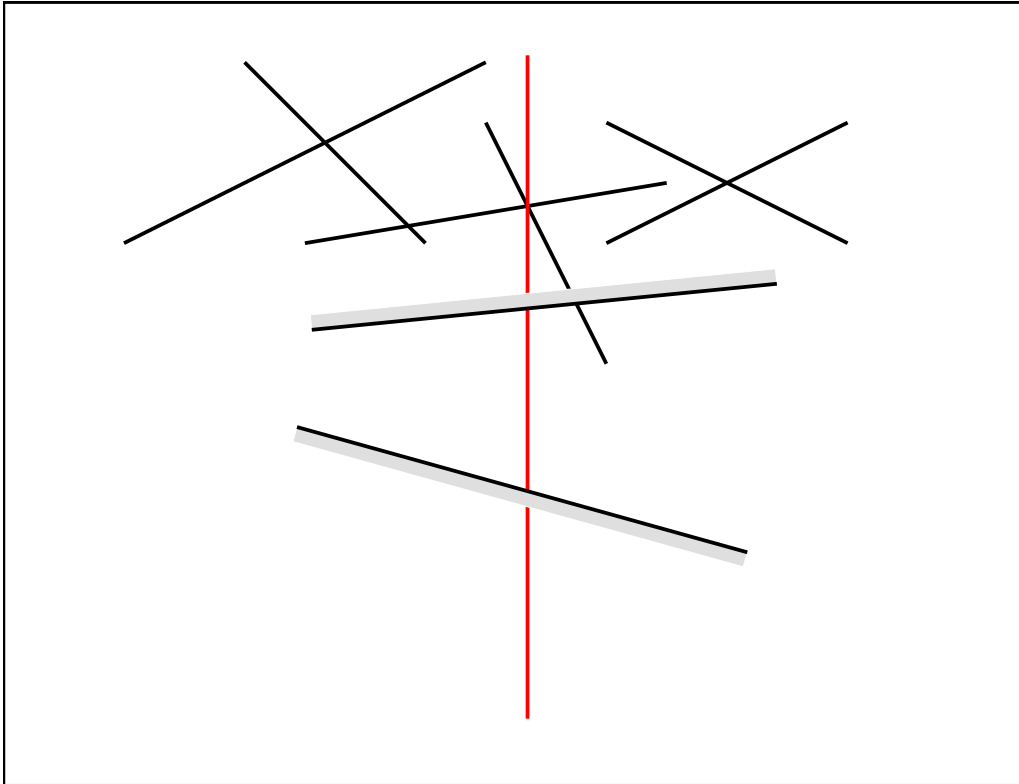
- Partition constraints into “down” and “up” constraints
- Pair “up-constraints” arbitrarily

Megiddo's Algorithm for LP in the plane



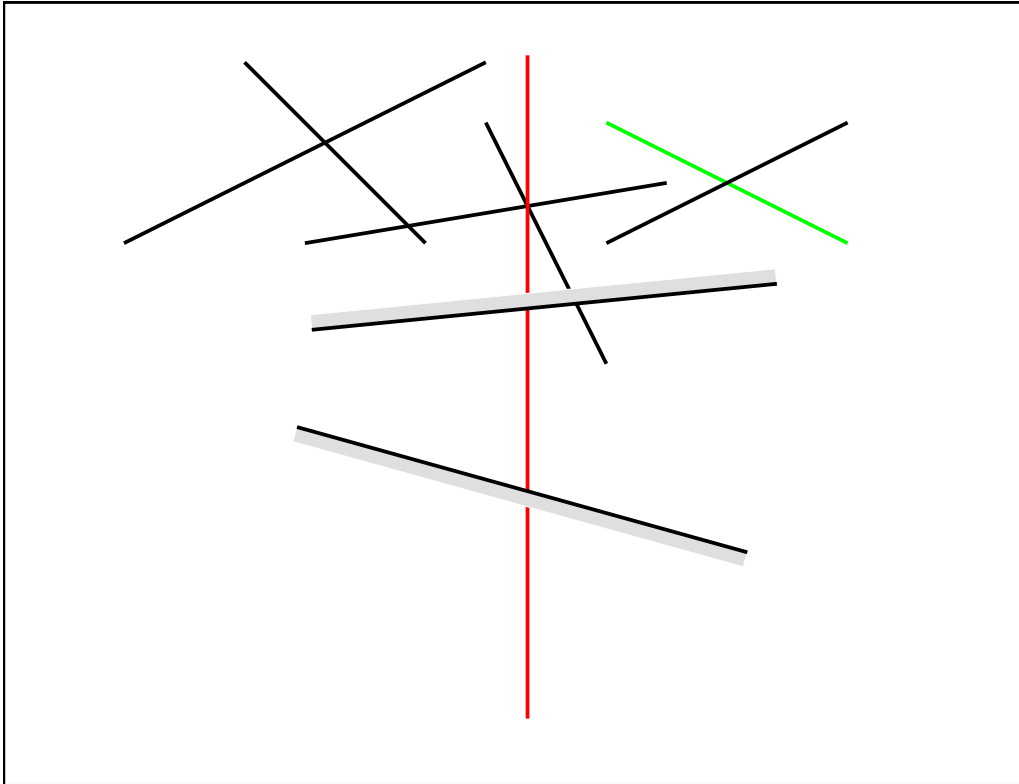
- Partition constraints into “down” and “up” constraints
- Pair “up-constraints” arbitrarily
- Compute median of intersections

Megiddo's Algorithm for LP in the plane



- Partition constraints into “down” and “up” constraints
- Pair “up-constraints” arbitrarily
- Compute median of intersections
- Decide whether optimum is left or right

Megiddo's Algorithm for LP in the plane



- Partition constraints into “down” and “up” constraints
- Pair “up-constraints” arbitrarily
- Compute median of intersections
- Decide whether optimum is left or right
- Prune 1/4-th of constraints

Megiddo's Algorithm for LP in the plane

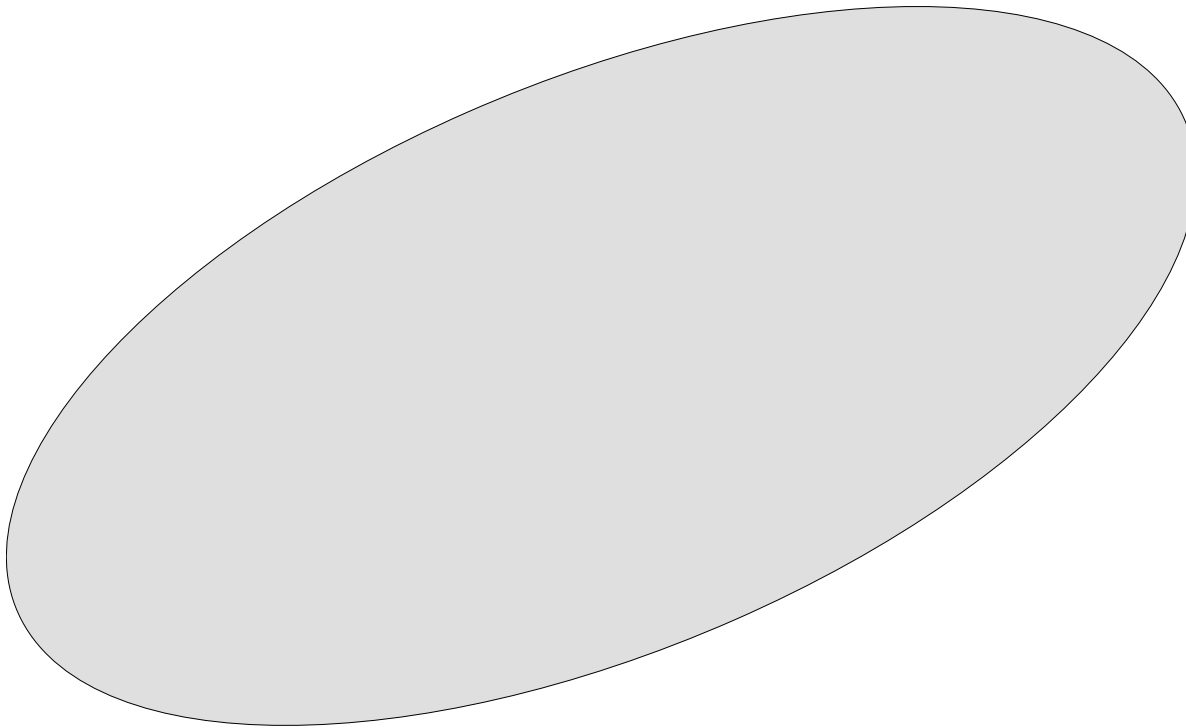
- Each round at least 1/4-th of the constraints pruned
- Each round costs linear time
- Overall cost is linear

Theorem ([Meg83]). *A linear program in the plane with m constraints can be solved in $O(m)$.*

Combining Prune&Search with feasibility algorithm

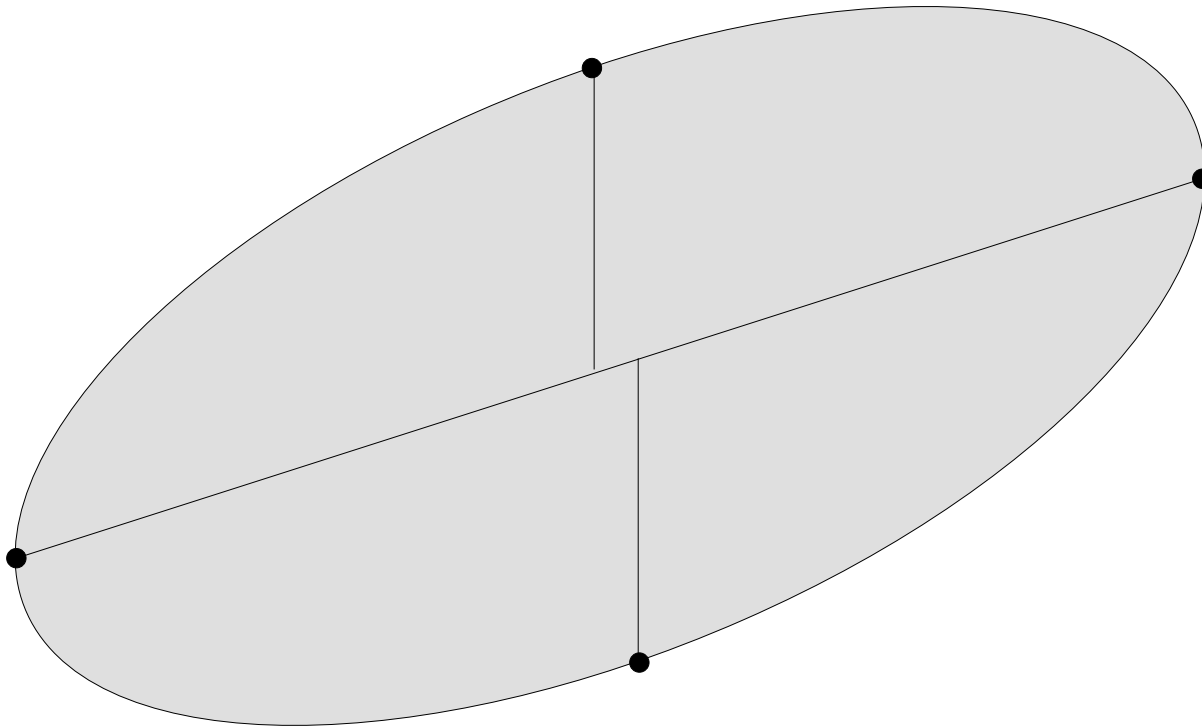
Partitioning the Polygon

$x(1)$

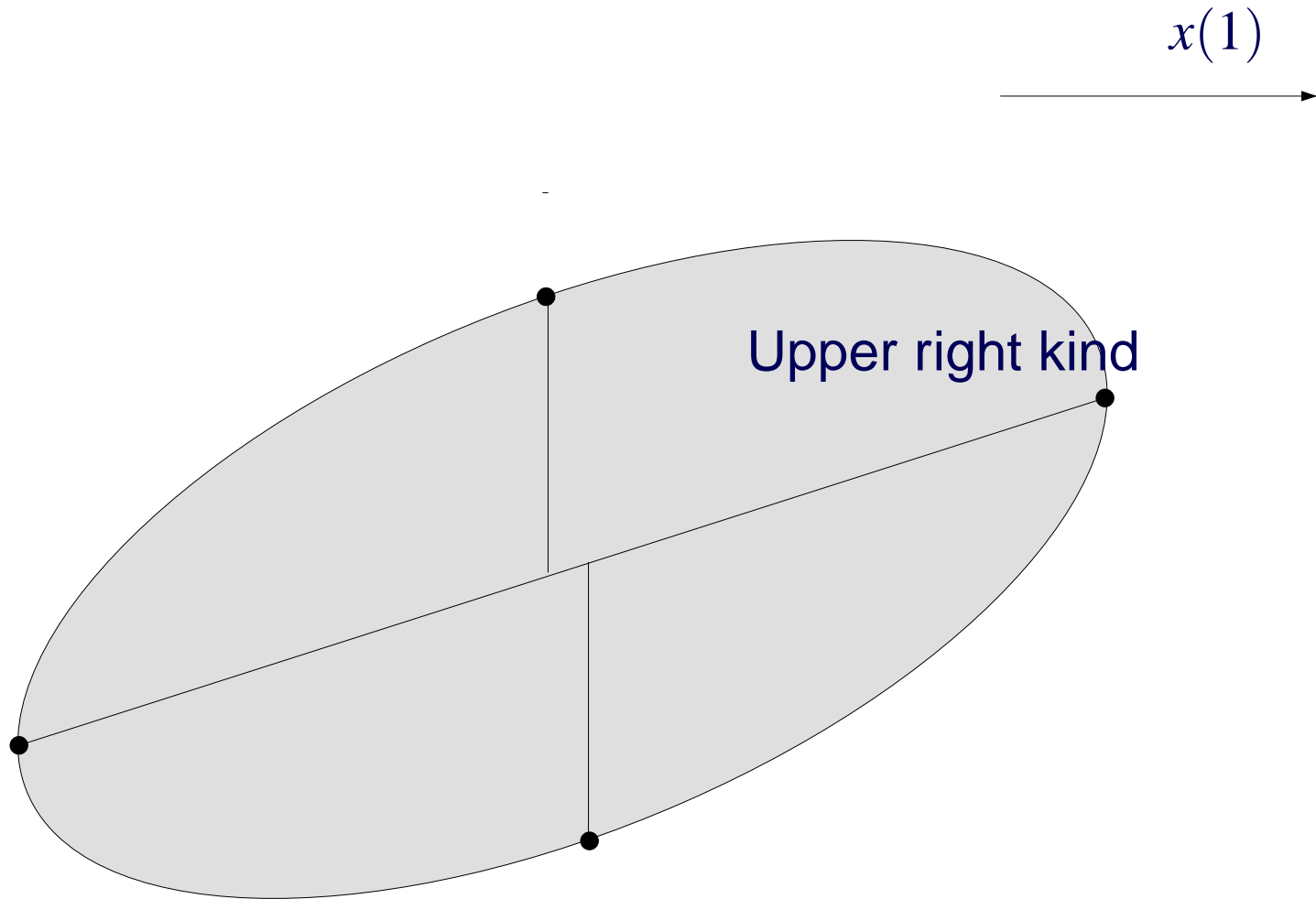


Partitioning the Polygon

$x(1)$

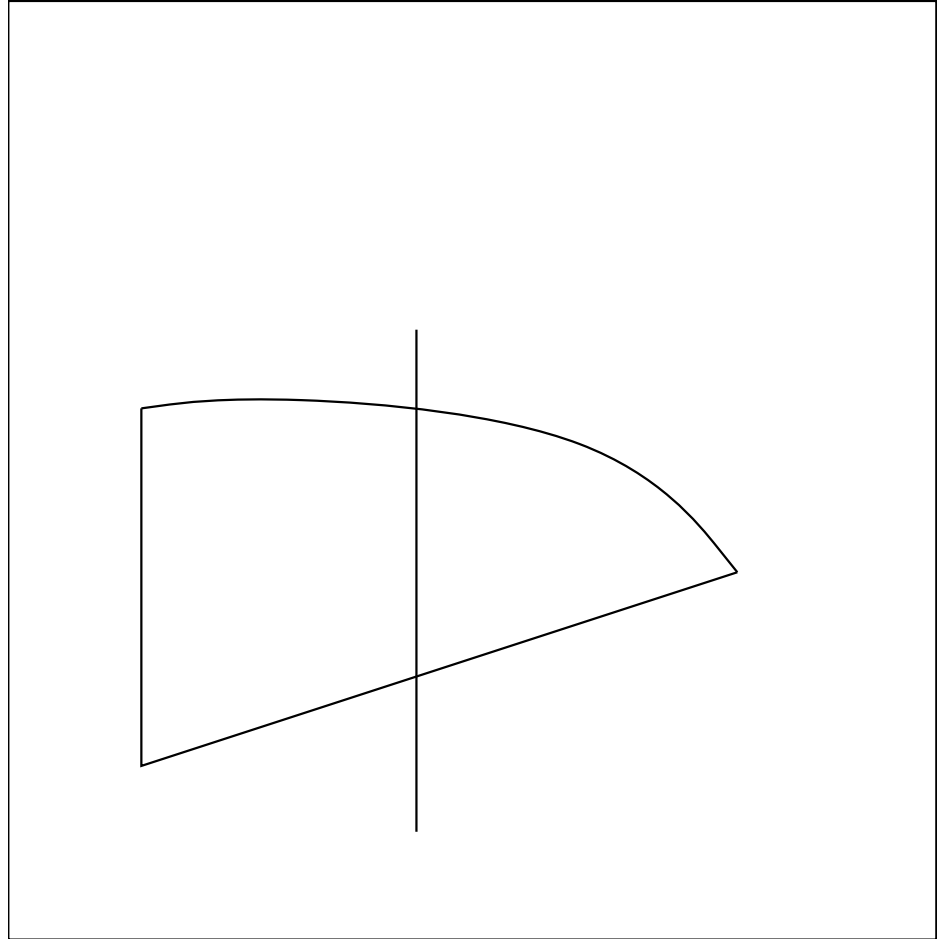


Partitioning the Polygon



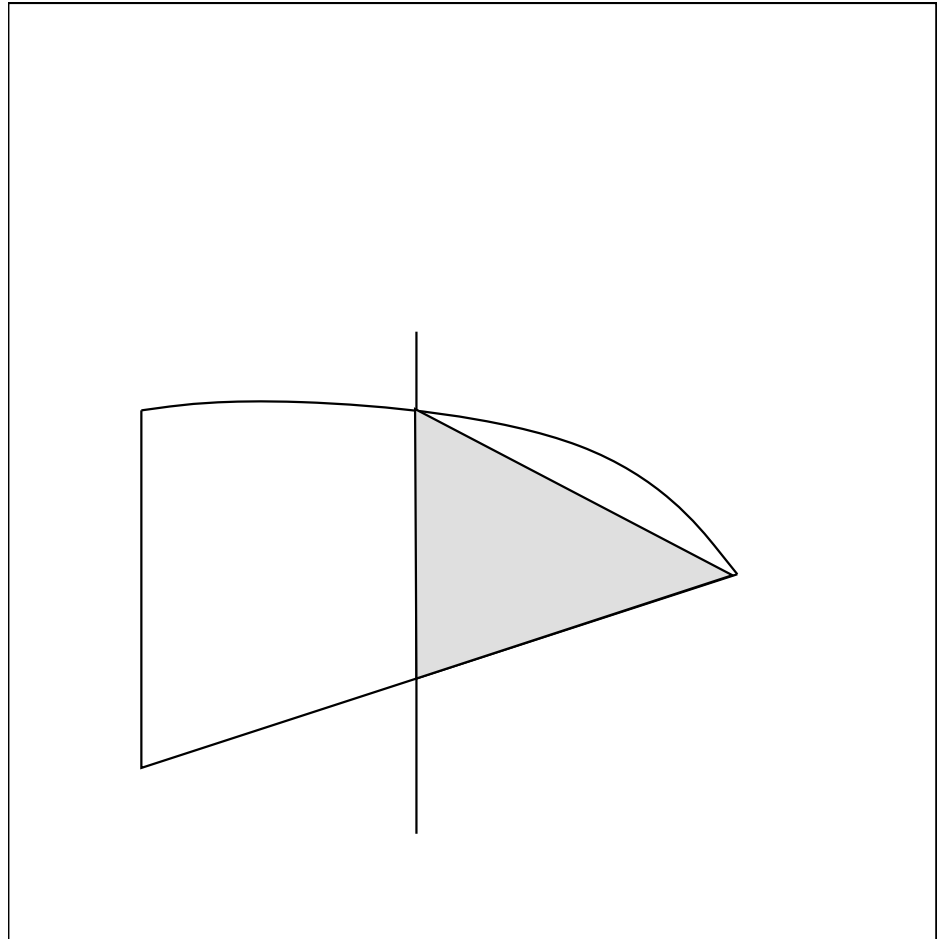
Upper right kind

- Consider $P \cap x(1) \geq \ell$



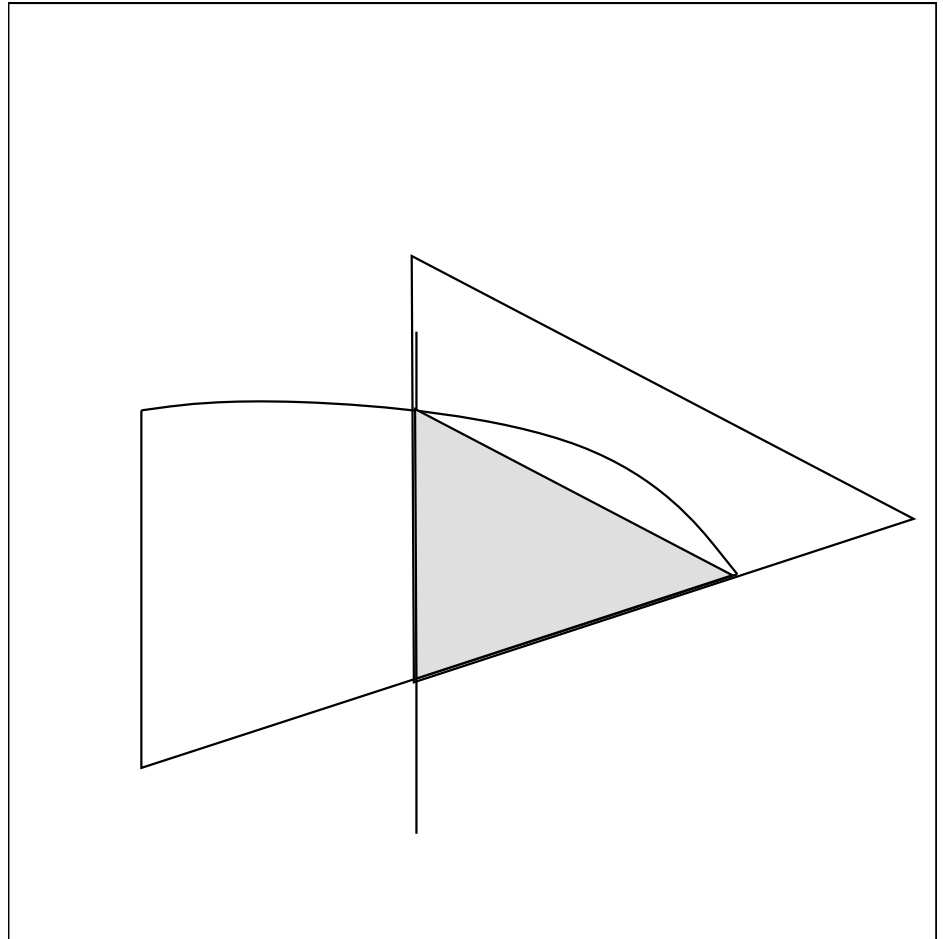
Upper right kind

- Consider $P \cap x(1) \geq \ell$
- Width of triangle is about width of $P \cap x(1) \geq \ell$



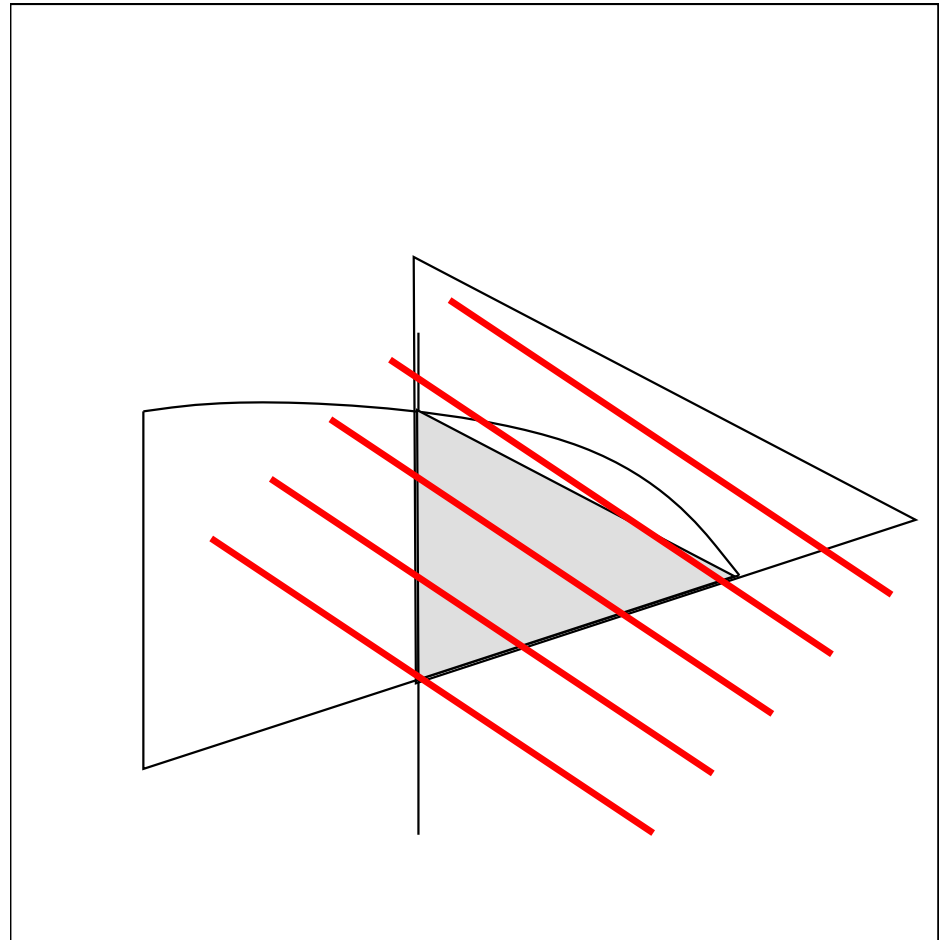
Upper right kind

- Consider $P \cap x(1) \geq \ell$
- Width of triangle is about width of $P \cap x(1) \geq \ell$
- Determine position ℓ , for which width of triangle is $f_2 + \varepsilon$



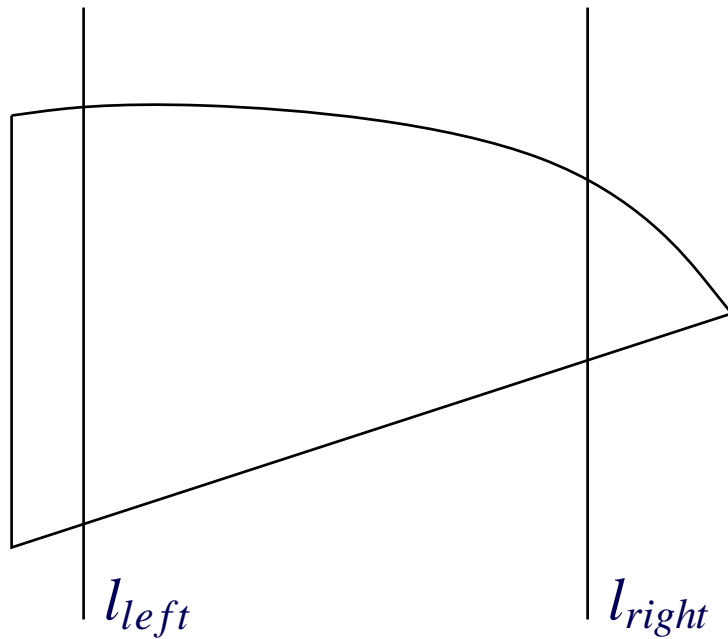
Upper right kind

- Consider $P \cap x(1) \geq \ell$
- Width of triangle is about width of $P \cap x(1) \geq \ell$
- Determine position ℓ , for which width of triangle is $f_2 + \varepsilon$
- Reduce problem to a constant number of problems on the line

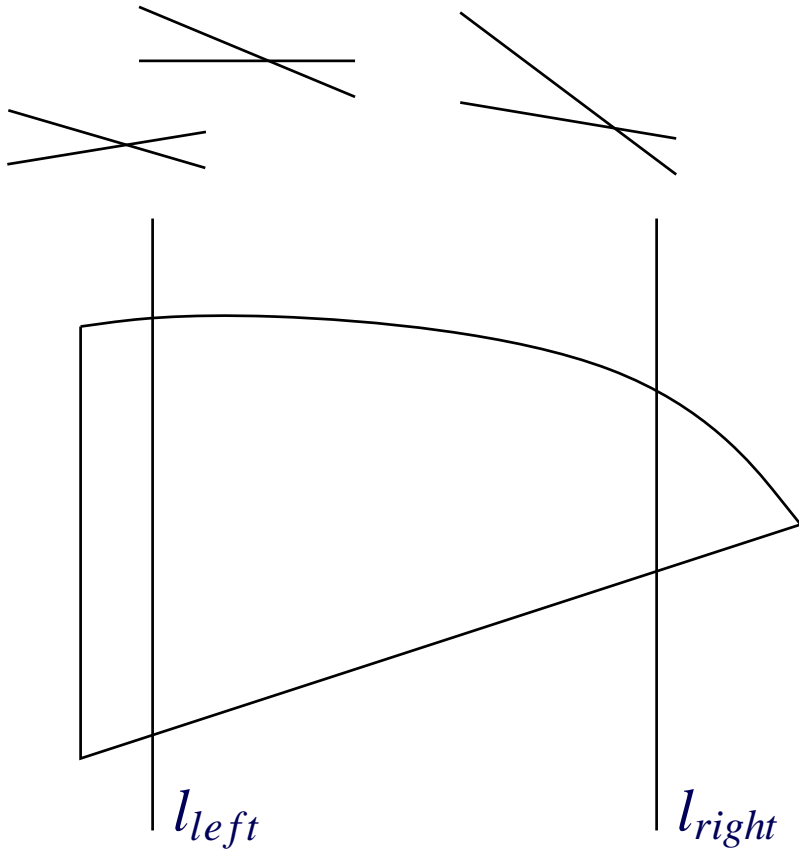


Prune & Search

- Principle: Improve l_{left} and l_{right}

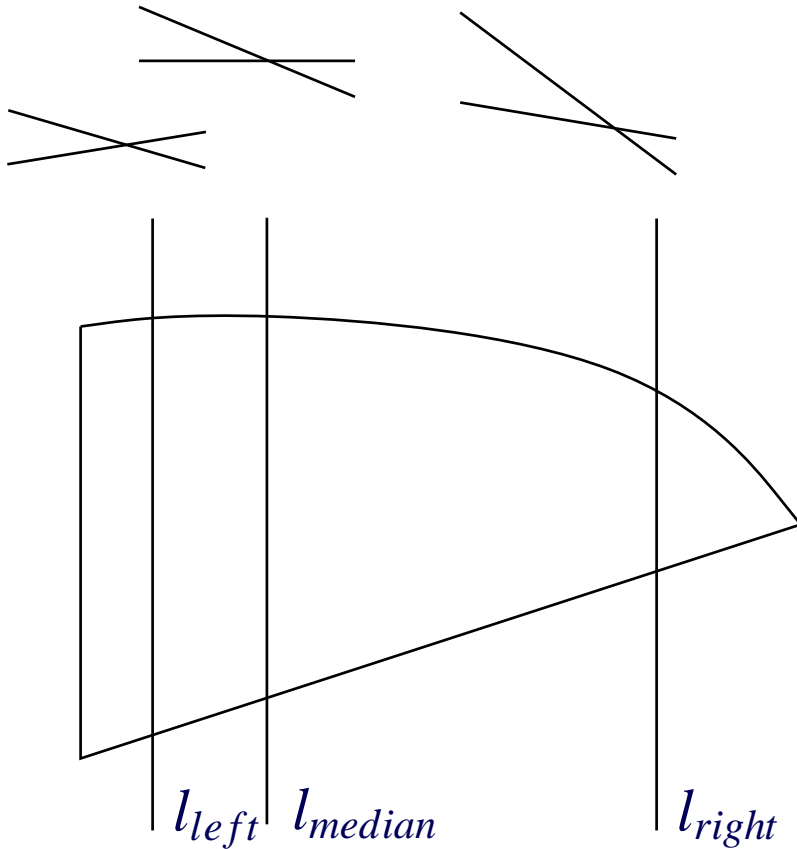


Prune & Search



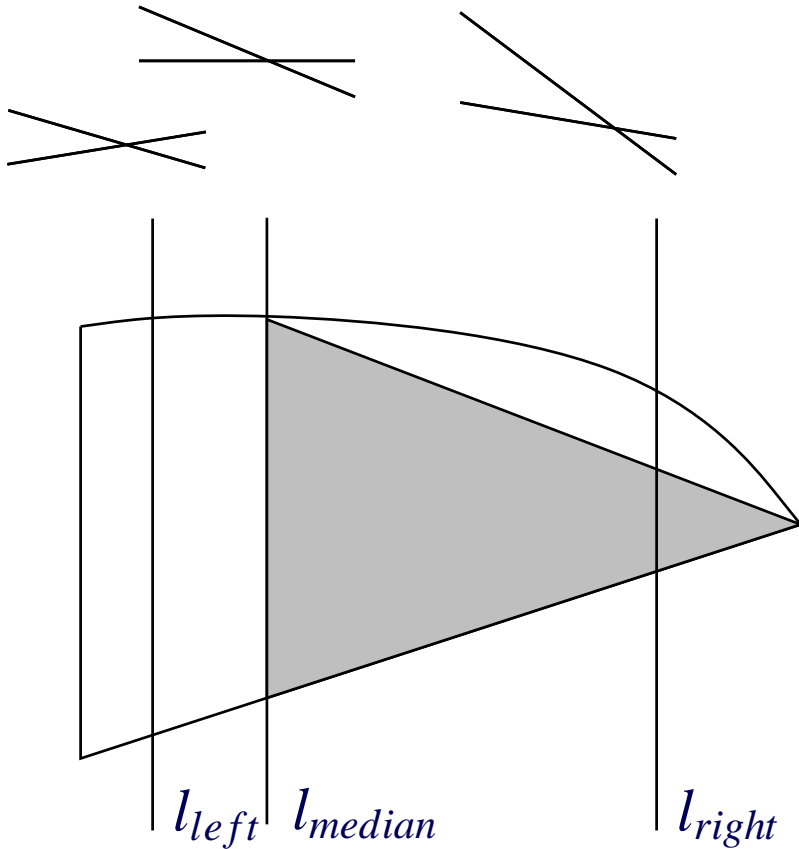
- Principle: Improve l_{left} and l_{right}
- Pair constraints arbitrarily

Prune & Search



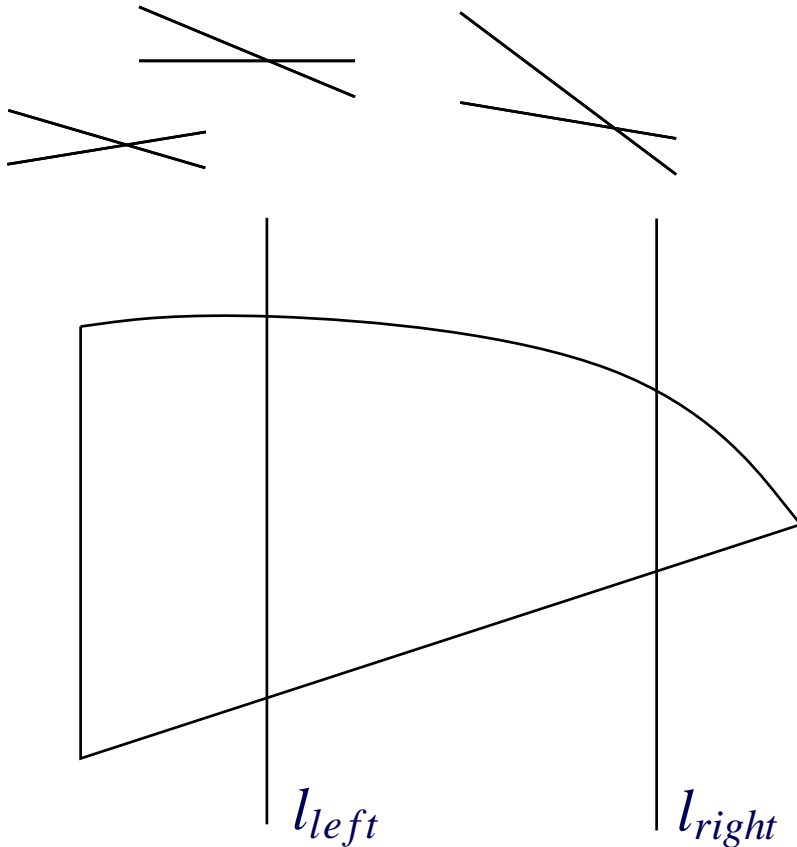
- Principle: Improve l_{left} and l_{right}
- Pair constraints arbitrarily
- Compute median of intersections

Prune & Search



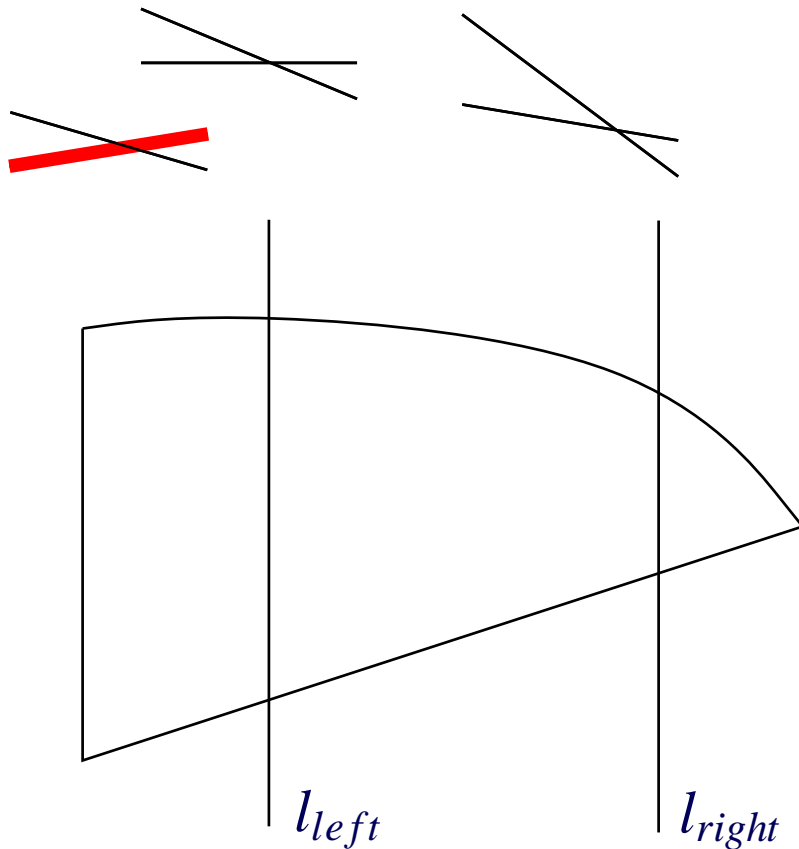
- Principle: Improve l_{left} and l_{right}
- Pair constraints arbitrarily
- Compute median of intersections
- Compute width of triangle defined by median

Prune & Search



- Principle: Improve l_{left} and l_{right}
- Pair constraints arbitrarily
- Compute median of intersections
- Compute width of triangle defined by median
- Update bounds

Prune & Search



- Principle: Improve l_{left} and l_{right}
- Pair constraints arbitrarily
- Compute median of intersections
- Compute width of triangle defined by median
- Update bounds
- Prune 1/4-th of constraints

Analysis

- Each round $1/4$ -th of constraints pruned

Analysis

- Each round $1/4$ -th of constraints pruned
- Computing median is linear

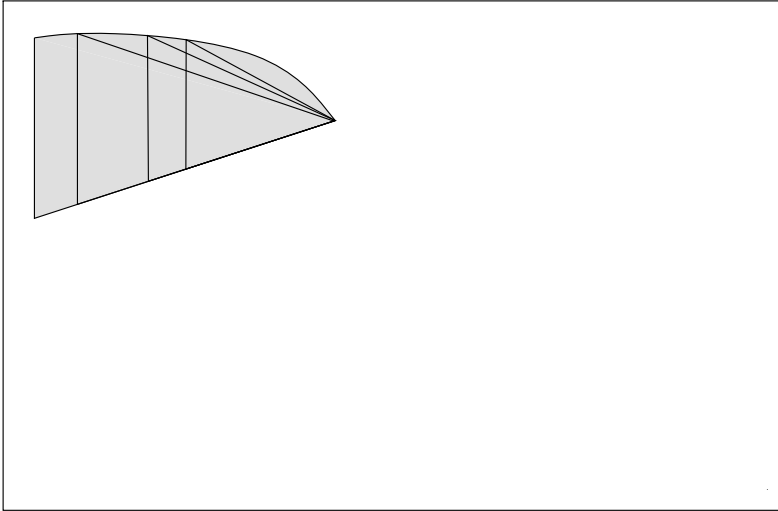
Analysis

- Each round 1/4-th of constraints pruned
- Computing median is linear
- Running time without width checking: $O(m)$

Analysis

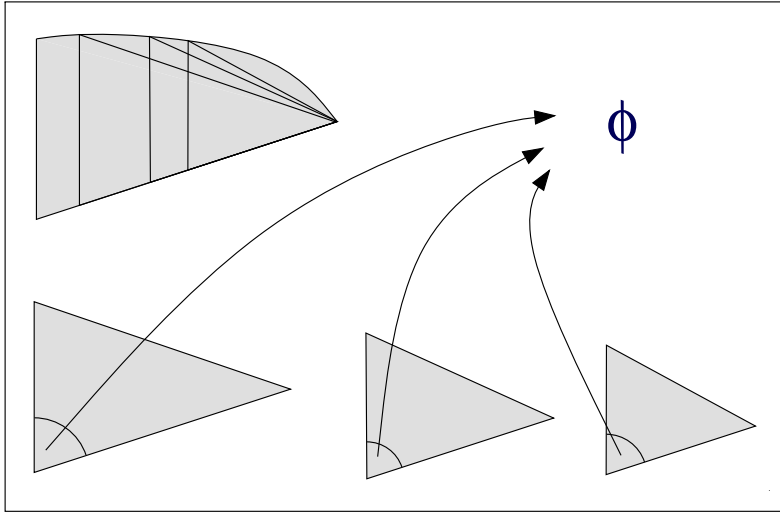
- Each round 1/4-th of constraints pruned
- Computing median is linear
- Running time without width checking: $O(m)$
- Number of checked triangles: $O(\log m)$

The checked triangles



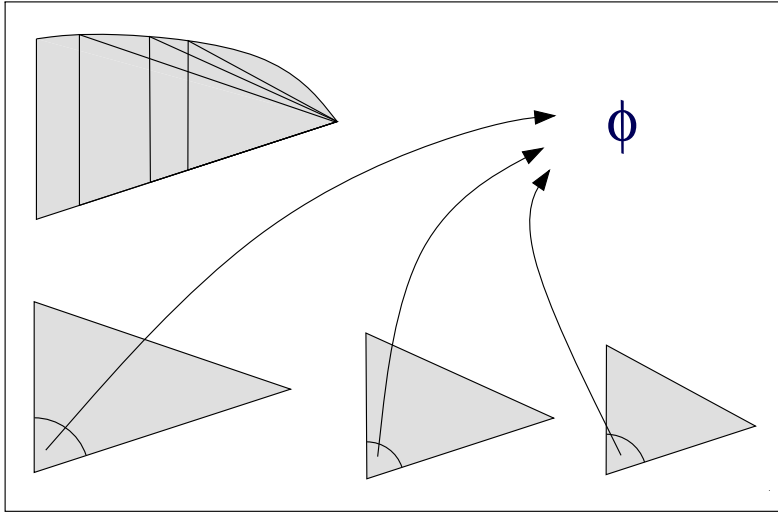
- Let $\begin{pmatrix} u^T \\ v^T \end{pmatrix}$ be the matrix of one prototype triangle

The checked triangles



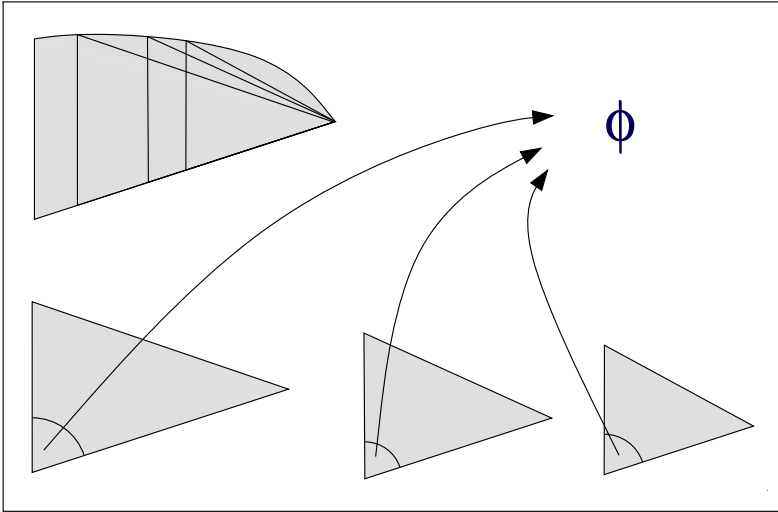
- Let $\begin{pmatrix} u^T \\ v^T \end{pmatrix}$ be the matrix of one prototype triangle
- Query: Given $\alpha \in \mathbb{Q}$, $\beta \in \mathbb{Q}$ compute shortest vector of lattice $\Lambda = \left\{ \alpha \begin{pmatrix} u^T \\ \beta v^T \end{pmatrix} x \mid x \in \mathbb{Z}^2 \right\}$

The checked triangles



- Let $\begin{pmatrix} u^T \\ v^T \end{pmatrix}$ be the matrix of one prototype triangle
- Query: Given $\alpha \in \mathbb{Q}$, $\beta \in \mathbb{Q}$ compute shortest vector of lattice $\Lambda = \{ \alpha \begin{pmatrix} u^T \\ \beta v^T \end{pmatrix} x \mid x \in \mathbb{Z}^2 \}$
- α can be neglected

The checked triangles



- Let $\begin{pmatrix} u^T \\ v^T \end{pmatrix}$ be the matrix of one prototype triangle
- Query: Given $\alpha \in \mathbb{Q}$, $\beta \in \mathbb{Q}$ compute shortest vector of lattice $\Lambda = \{ \alpha \begin{pmatrix} u^T \\ \beta v^T \end{pmatrix} x \mid x \in \mathbb{Z}^2 \}$
- α can be neglected

QUERY:

Given $\beta \in \mathbb{Q}$, compute shortest vector of lattice

$$\Lambda = \left\{ \begin{pmatrix} u^T \\ \beta v^T \end{pmatrix} x \mid x \in \mathbb{Z}^2 \right\}$$

Batching the width checks

Theorem. Shortest vector of $\Lambda = \left\{ \begin{pmatrix} a & b \\ 0 & \beta c \end{pmatrix} x \mid x \in \mathbb{Z}^2 \right\}$ is $\begin{pmatrix} -xa+yc \\ yb \end{pmatrix}$, where x/y convergent of b/a .

- Preprocessing: Compute list of convergents $x(1)/y(1), \dots, x(k)/y(k)$ of b/a
- Complexity: $O(s)$

Batching the width checks

Incoming query: $\Lambda = \left\{ \begin{pmatrix} a & b \\ 0 & \beta c \end{pmatrix} x \mid x \in \mathbb{Z}^2 \right\}$

- Search convergent $x(j)/y(j)$ with minimal $\max\{|-x(j)a + y(j)b|, |\beta y(j)c|\}$

Batching the width checks

Incoming query: $\Lambda = \left\{ \begin{pmatrix} a & b \\ 0 & \beta c \end{pmatrix} x \mid x \in \mathbb{Z}^2 \right\}$

- Search convergent $x(j)/y(j)$ with minimal $\max\{|-x(j)a + y(j)b|, |\beta y(j)c|\}$
- Sequence $|-x(j)a + y(j)b|$ is decreasing

Batching the width checks

Incoming query: $\Lambda = \left\{ \begin{pmatrix} a & b \\ 0 & \beta c \end{pmatrix} x \mid x \in \mathbb{Z}^2 \right\}$

- Search convergent $x(j)/y(j)$ with minimal $\max\{|-x(j)a + y(j)b|, |\beta y(j)c|\}$
- Sequence $|-x(j)a + y(j)b|$ is decreasing
- Sequence $|\beta y(j)c|$ is increasing

Batching the width checks

Incoming query: $\Lambda = \left\{ \begin{pmatrix} a & b \\ 0 & \beta c \end{pmatrix} x \mid x \in \mathbb{Z}^2 \right\}$

- Search convergent $x(j)/y(j)$ with minimal $\max\{|-x(j)a + y(j)b|, |\beta y(j)c|\}$
- Sequence $|-x(j)a + y(j)b|$ is decreasing
- Sequence $|\beta y(j)c|$ is increasing
- Binary search: One query costs $O(\log(s))$

Batching the width checks

Incoming query: $\Lambda = \left\{ \begin{pmatrix} a & b \\ 0 & \beta c \end{pmatrix} x \mid x \in \mathbb{Z}^2 \right\}$

- Search convergent $x(j)/y(j)$ with minimal $\max\{|-x(j)a + y(j)b|, |\beta y(j)c|\}$
- Sequence $|-x(j)a + y(j)b|$ is decreasing
- Sequence $|\beta y(j)c|$ is increasing
- Binary search: One query costs $O(\log(s))$
- Preprocessing and $O(\log m)$ queries: $O(s + \log m \cdot \log s)$

Batching the width checks

Incoming query: $\Lambda = \left\{ \begin{pmatrix} a & b \\ 0 & \beta c \end{pmatrix} x \mid x \in \mathbb{Z}^2 \right\}$

- Search convergent $x(j)/y(j)$ with minimal $\max\{|-x(j)a + y(j)b|, |\beta y(j)c|\}$
- Sequence $|-x(j)a + y(j)b|$ is decreasing
- Sequence $|\beta y(j)c|$ is increasing
- Binary search: One query costs $O(\log(s))$
- Preprocessing and $O(\log m)$ queries: $O(s + \log m \cdot \log s)$
- With prune & search $O(m + s)$

Total complexity

Theorem (E. & Laue). *IP in the plane can be solved in $O(m + s)$.*

Research problems

- Is there a deterministic $O(m + s \log m)$ algorithm ?
- Is there a $O(m + s)$ algorithm ?

Bibliography

References

- [Ban96] W. Banaszczyk. Inequalities for convex bodies and polar reciprocal lattices in \mathbf{R}^n . II. Application of K -convexity. *Discrete Comput. Geom.*, 16(3):305–311, 1996.
- [BLPS99] Wojciech Banaszczyk, Alexander E. Litvak, Alain Pajor, and Stanislaw J. Szarek. The flatness theorem for nonsymmetric convex bodies via the local theory of Banach spaces. *Mathematics of Operations Research*, 24(3):728–750, 1999.
- [EL05] F. Eisenbrand and S. Laue.
A linear algorithm for integer programming in the plane. *Mathematical Programming*, 102(2):249 – 259, 2005.
- [Gau01] C. F. Gauß. *Disquisitiones arithmeticae*. Gerh. Fleischer Iun., 1801.
- [Her50] Ch. Hermite. Extraits de lettres de M. Ch. Hermite à M. Jacobi sur différents objets de la théorie des nombres. *Journal für die reine und angewandte Mathematik*, 40, 1850.
- [Len83] H. W. Lenstra. Integer programming with a fixed number of variables. *Mathematics of Operations Research*, 8(4):538 – 548, 1983.
- [LLL82] A. K. Lenstra, H. W. Lenstra, and L. Lovász. Factoring polynomials with rational coefficients. *Math. Annalen*, 261:515 – 534, 1982.