Efficient Algorithms for Integer Programming

Research Problem

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

What is integer programming

- Variables: $x(1), \ldots, x(n)$
- Linear constraints: $a_{i1}x(1) + \cdots + a_{in}x(n) \leqslant b(i)$, for $i = 1, \ldots, m$
- Linear objective function: $c(1)x(1) + \cdots + c(n)x(n)$

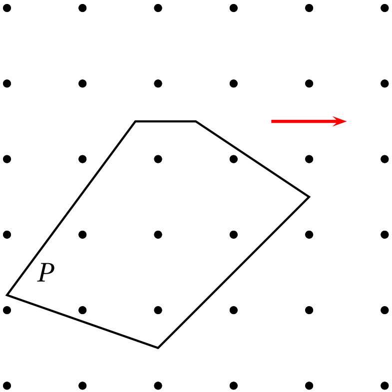
- Task: Find integer assignment to $x(1), \ldots, x(n)$ such that all constraints are satisfied and objective function is maximized.

Geometric interpretation

- Given a (bounded) Polyhedron $P = \{x \in \mathbb{R}^n \mid Ax \leqslant b\}$
- Find vertex of the integer hull P_I of P which maximizes objective function $c^T x$

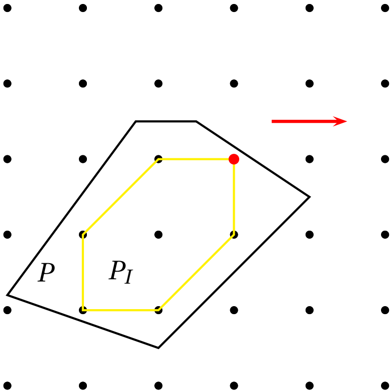
Geometric interpretation

- Given a (bounded) Polyhedron $P = \{x \in \mathbb{R}^n \mid Ax \leqslant b\}$
- Find vertex of the integer hull P_I of P which maximizes objective function $c^T x$



Geometric interpretation

- Given a (bounded) Polyhedron $P = \{x \in \mathbb{R}^n \mid Ax \leqslant b\}$
- Find vertex of the integer hull P_I of P which maximizes objective function $c^T x$



A first simple result in fixed dimension

How many extreme points (vertices) can P_I have?

A first simple result in fixed dimension

How many extreme points (vertices) can P_I have?
Consider a Knapsack Polyhedron defined by integral data

$$a(1)x(1) + \cdots + a(n)x(n) \leqslant \beta, \quad x \geqslant 0$$

And two different vertices of P_I

$$(x(1), \ldots, x(n)) \quad \text{and} \quad (y(1), \ldots, y(n))$$

and suppose that $\lfloor \log(x(i)) \rfloor = \lfloor \log(y(i)) \rfloor$ for $i = 1, \ldots, n$.

A first simple result in fixed dimension

How many extreme points (vertices) can P_I have?
Consider a Knapsack Polyhedron defined by integral data

$$a(1)x(1) + \cdots + a(n)x(n) \leqslant \beta, \quad x \geqslant 0$$

And two different vertices of P_I

$$(x(1), \ldots, x(n)) \quad \text{and} \quad (y(1), \ldots, y(n))$$

and suppose that $\lfloor \log(x(i)) \rfloor = \lfloor \log(y(i)) \rfloor$ for $i = 1, \ldots, n$.
Then

- $2 \cdot x - y \geqslant 0$ and $2 \cdot y - x \geqslant 0$
- $a^T \big((2 \cdot x - y) + (2 \cdot y - x) \big) = a^T (x + y) \leqslant 2 \cdot \beta$

W.l.o.g. one can assume that $a^T (2 \cdot x - y) \leqslant \beta$.
But then $1/2(2 \cdot x - y) + 1/2 \cdot y = x$ which contradicts that x is a vertex.

The number of extreme points is polynomial

- Consider simplex with vertex 0

$$S = \{x \in \mathbb{R}^n \mid Bx \geqslant 0,\ a^T x \leqslant \beta\}$$

  with $B \in \mathbb{Z}^{n\times n}$ invertible.
- $S = \{x \in \mathbb{R}^n \mid Bx \geqslant 0,\ (B^{-1}a)^T(Bx) \leqslant \beta\}$
- $x \in \mathbb{Z}^n$ is vertex of S_I if and only if Bx is vertex of $\operatorname{conv}(K \cap \Lambda(B))$ with

$$K = \{x \in \mathbb{R}^n \mid x \geqslant 0,\ (B^{-1}a)^T x \leqslant \beta\}$$

Exercise

Show that the number of vertices of $\operatorname{conv}(K \cap \Lambda(B))$ is polynomial in fixed dimension.

The number of extreme points is polynomial

By triangulation of P:

Theorem (Hayes & Larman 1983, Schrijver 1986)

Let $Ax \leqslant b$ be an integral system of inequalities, where $A \in \mathbb{Z}^{m\times n}$ and $b \in \mathbb{Z}^m$ and n is fixed. The integer hull P_I of $P = \{x \in \mathbb{R}^n \mid Ax \leqslant b\}$ has a polynomial number of extreme points.

polynomial in binary encoding length of A and b

The number of extreme points is polynomial

By triangulation of P:

Theorem (Hayes & Larman 1983, Schrijver 1986)

Let $Ax \leqslant b$ be an integral system of inequalities, where $A \in \mathbb{Z}^{m \times n}$ and $b \in \mathbb{Z}^m$ and n is fixed. The integer hull P_I of $P = \{x \in \mathbb{R}^n \mid Ax \leqslant b\}$ has a polynomial number of extreme points. $O(m^n \cdot s^n)$

polynomial in binary encoding length of A and b

Tight bounds for simplices: Bárány, Howe & Lovász 1992

Cook, Hartmann, Kannan & McDiarmid 1992

Polynomial algorithms for IP in fixed dimension

GCDs and IP

Theorem

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

$$\begin{array}{ll} \textit{minimize} & xa + yb \\ \textit{condition} & xa + yb \geqslant 1 \\ & x, y \in \mathbb{Z}. \end{array}$$

GCDs and IP

Theorem

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

$$\begin{array}{ll} \textit{minimize} & xa + yb \\ \textit{condition} & xa + yb \geqslant 1 \\ & x, y \in \mathbb{Z}. \end{array}$$

Integer Programming: **Combinatorics & Number Theory**

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

Theorem (Lenstra 1983)

The IP feasibility problem can be solved in polynomial time in fixed dimension.

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

Theorem (Lenstra 1983)

The IP feasibility problem can be solved in polynomial time in fixed dimension.

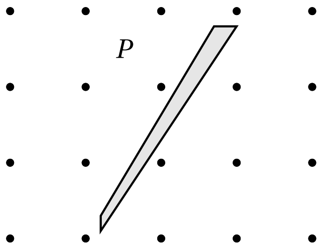
- $O(m+s)$ for feasibility
- $O(s \cdot (m+s))$ for optimization

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 $\leqslant$ constant f_n

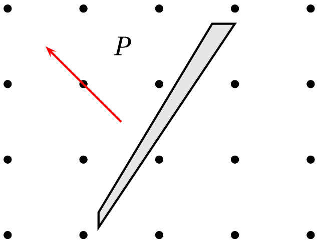


Flatness theorem

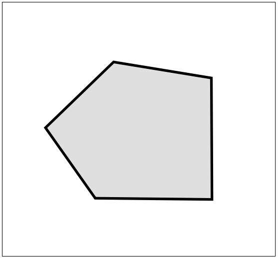
Width of P along c, $w_c(P)$: $\max\{c^T x \mid x \in P\} - \min\{c^T x \mid x \in P\}$

Theorem (Khinchine's flatness theorem)

If $P_I = \emptyset$, then there exists integral $c \neq 0$ such that width of P along c is $\leqslant$ constant f_n

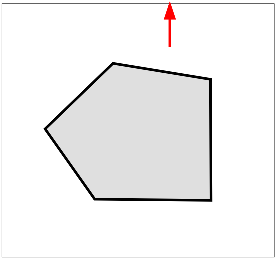


Lenstra's IP algorithm



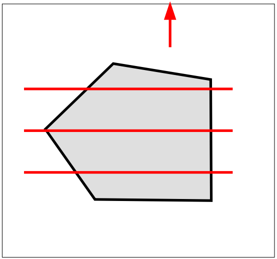
- Lenstra's algorithm is an algorithm for IP feasibility
- Computes width of polyhedron
- If width is to large, then return **feasible**
- Otherwise, **recursively** search for integer point on one of the constant number of hyperplanes (lower dimension)

Lenstra's IP algorithm



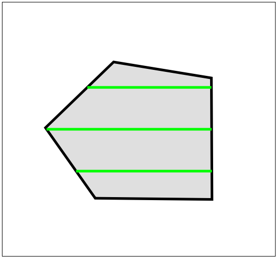
- Lenstra's algorithm is an algorithm for IP feasibility
- Computes width of polyhedron
- If width is to large, then return **feasible**
- Otherwise, **recursively** search for integer point on one of the constant number of hyperplanes (lower dimension)

Lenstra's IP algorithm



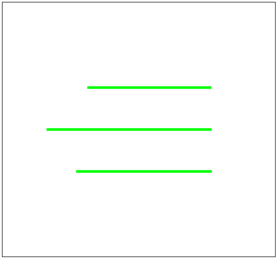
- Lenstra's algorithm is an algorithm for IP feasibility
- Computes width of polyhedron
- If width is to large, then return **feasible**
- Otherwise, **recursively** search for integer point on one of the constant number of hyperplanes (lower dimension)

Lenstra's IP algorithm



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

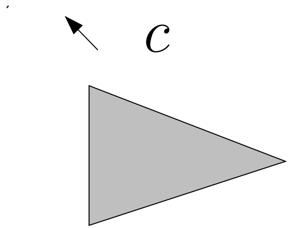
Lenstra’s IP algorithm



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

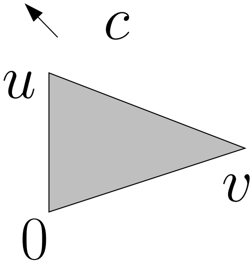
Width of a triangle

- W.l.o.g. 0 is a vertex



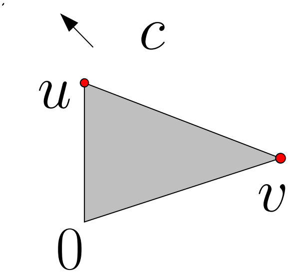
Width of a triangle

- W.l.o.g. 0 is a vertex
- Width along c is $\max\{c^T u, c^T v, 0\} - \min\{c^T u, c^T v, 0\}$



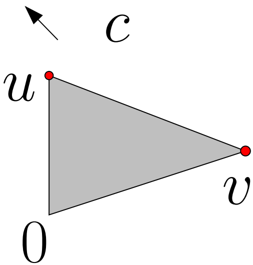
Width of a triangle

- W.l.o.g. 0 is a vertex
- Width along c is $\max\{c^T u, c^T v, 0\} - \min\{c^T u, c^T v, 0\}$
- $|a-b| \leqslant |a| + |b| \implies$ width along c $\leqslant 2\max\{|c^T u|, |c^T v|\} = 2\left\|\begin{pmatrix} u^T \\ v^T \end{pmatrix} c\right\|_\infty$



Width of a triangle

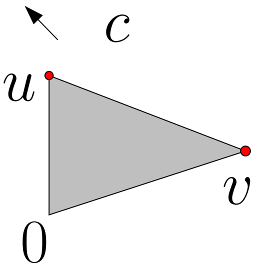
- W.l.o.g. 0 is a vertex
- Width along c is $\max\{c^T u, c^T v, 0\} - \min\{c^T u, c^T v, 0\}$
- $|a-b| \leqslant |a| + |b| \implies$ width along c $\leqslant 2\max\{|c^T u|, |c^T v|\} = 2\,\|\binom{u^T}{v^T} c\|_\infty$
- $\geqslant \max\{|c^T u|, |c^T v|\} = \|\binom{u^T}{v^T} c\|_\infty$



Width of a triangle

- W.l.o.g. 0 is a vertex
- Width along c is $\max\{c^T u, c^T v, 0\} - \min\{c^T u, c^T v, 0\}$
- $|a-b| \leqslant |a| + |b| \implies$ width along c $\leqslant 2\max\{|c^T u|, |c^T v|\} = 2\,\|\binom{u^T}{v^T}c\|_\infty$
- $\geqslant \max\{|c^T u|, |c^T v|\} = \|\binom{u^T}{v^T}c\|_\infty$
- Width of triangle $\approx$ length of shortest vector w.r.t. ℓ_∞ of lattice

$$\Lambda = \{\binom{u^T}{v^T}x \mid x \in \mathbb{Z}^2\}.$$



Shortest vectors in dimension 2

A fraction x/y with $y \geqslant 1$ is a best approximation of $\alpha \in \mathbb{R}$ if $|y \cdot \alpha - x| \leqslant |y' \cdot \alpha - x'|$ for each fraction x'/y' with $1 \leqslant y' \leqslant y$.

Exercise

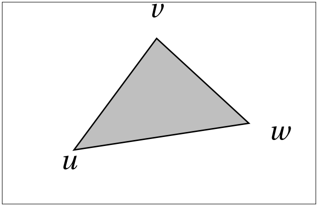
Consider the lattice $\Lambda = \left\{ \begin{pmatrix} a & b \\ 0 & c \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \mid \begin{pmatrix} x \\ y \end{pmatrix} \in \mathbb{Z}^2 \right\}$. Prove the following statement.

A shortest vector of Λ is either $\begin{pmatrix} a \\ 0 \end{pmatrix}$, $\begin{pmatrix} b \\ c \end{pmatrix}$ or is of the form $\begin{pmatrix} -x \cdot a + y \cdot b \\ y \cdot c \end{pmatrix}$, where x/y is a best approximation of $\alpha = b/a$.

Notice: The best approximations of a rational number can be computed in linear time with the Euclidean Algorithm

Deciding IP feasibility of triangles

- Given triangle $T = \mathrm{conv}\{u, v, w\}$

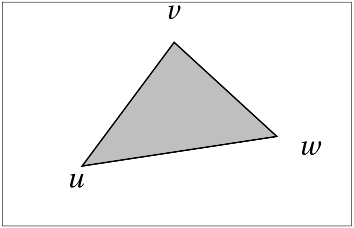


Exercise

How can we efficiently decide, whether a line-segment contains an integer point? *Hint: Consider the line on which the segment lies and apply a unimodular transformation with the help of the extended Euclidean algorithm.*

Deciding IP feasibility of triangles

- Given triangle $T = \mathrm{conv}\{u, v, w\}$
- Compute shortest vector $A \cdot c$, $c \in \mathbb{Z}^2$, where $A = \binom{v-u}{w-u}$

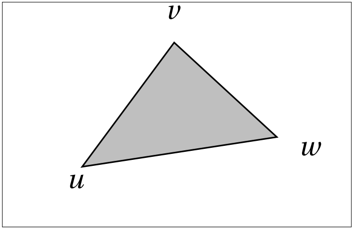


Exercise

How can we efficiently decide, whether a line-segment contains an integer point? *Hint: Consider the line on which the segment lies and apply a unimodular transformation with the help of the extended Euclidean algorithm.*

Deciding IP feasibility of triangles

- Given triangle $T = \operatorname{conv}\{u, v, w\}$
- Compute shortest vector $A \cdot c$, $c \in \mathbb{Z}^2$, where $A = \binom{v-u}{w-u}$
- If $\max\{c^T x \mid x \in T\} - \min\{c^T x \mid x \in T\} > f(2)$, then T feasible



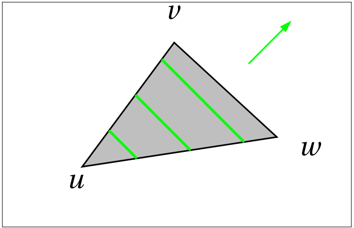
Exercise

How can we efficiently decide, whether a line-segment contains an integer point? *Hint: Consider the line on which the segment lies and apply a unimodular transformation with the help of the extended Euclidean algorithm.*

Deciding IP feasibility of triangles

- Given triangle $T = \text{conv}\{u, v, w\}$
- Compute shortest vector $A \cdot c$, $c \in \mathbb{Z}^2$, where $A = \binom{v-u}{w-u}$
- If $\max\{c^T x \mid x \in T\} - \min\{c^T x \mid x \in T\} > f(2)$, then T feasible
- Else decide feasibility of line segments

$$T \cap (c^T x = \delta),\ \delta \in \mathbb{Z}$$



Exercise

How can we efficiently decide, whether a line-segment contains an integer point? *Hint: Consider the line on which the segment lies and apply a unimodular transformation with the help of the extended Euclidean algorithm.*

Complexity

- Shortest vector computation in linear time
- Line segments can be checked in linear time
- IP feasibility of triangle decidable in linear time
- Integer feasibility of polygons can be decided in polynomial time via triangulation $O(m \cdot s)$

Effient algorithms for IP-optimization in the plane

IP in the plane: History

m: Number of constraints

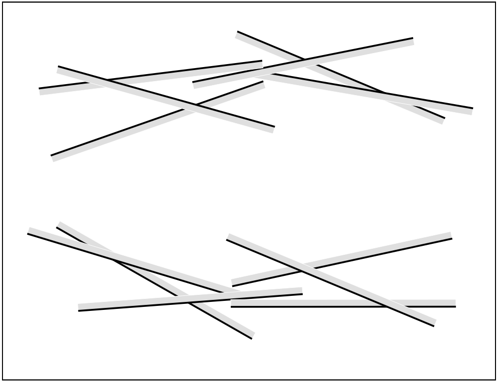
s: largest binary encoding length of coefficient

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 2004	$O(m + s)$
Feasibility test + Euclidean algorithm	$O(m + s)$

any fixed dimension

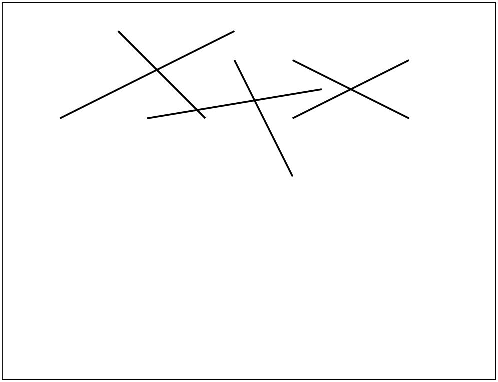
Prune & Search: Dealing with the combinatorics

Megiddo's Algorithm for LP in the plane



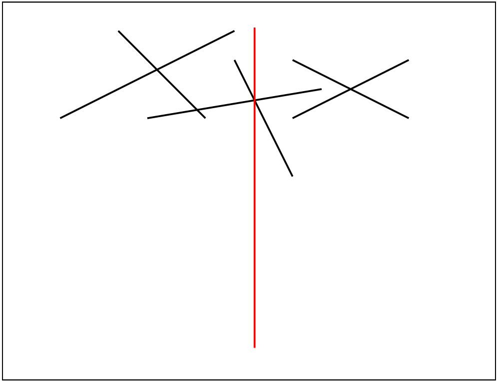
- Partition constraints into "down" and "up" constraints

Megiddo's Algorithm for LP in the plane



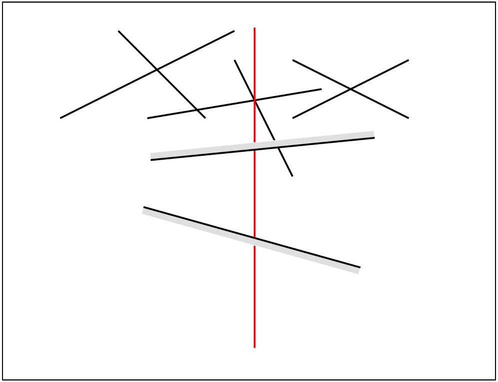
- Partition constraints into "down" and "up" constraints
- Pair "up-constraints" arbitrarily

Megiddo's Algorithm for LP in the plane



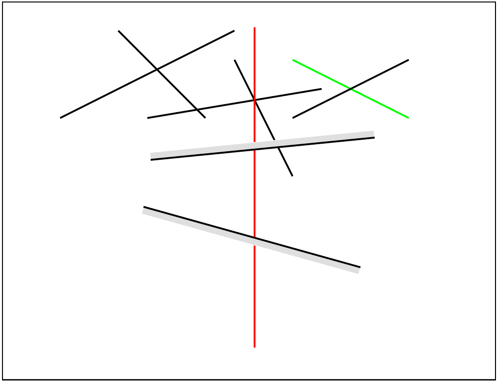
- Partition constraints into "down" and "up" constraints
- Pair "up-constraints" arbitrarily
- Compute median of intersections

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

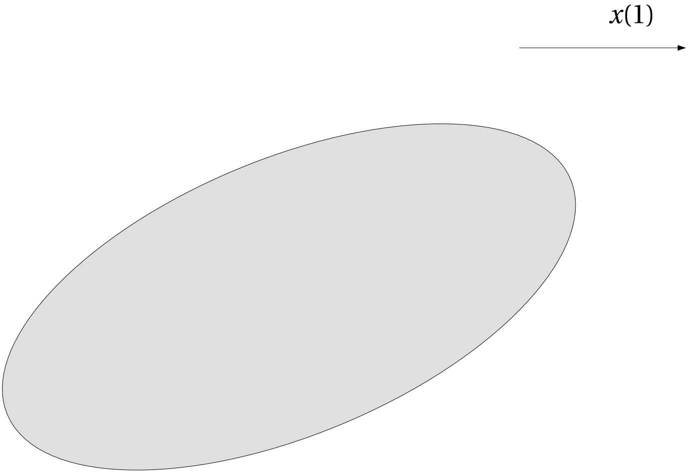
- Each round at least 1/4-th of the constraints pruned
- Each round costs linear time
- Overall cost is linear

Theorem (Megiddo 1983)

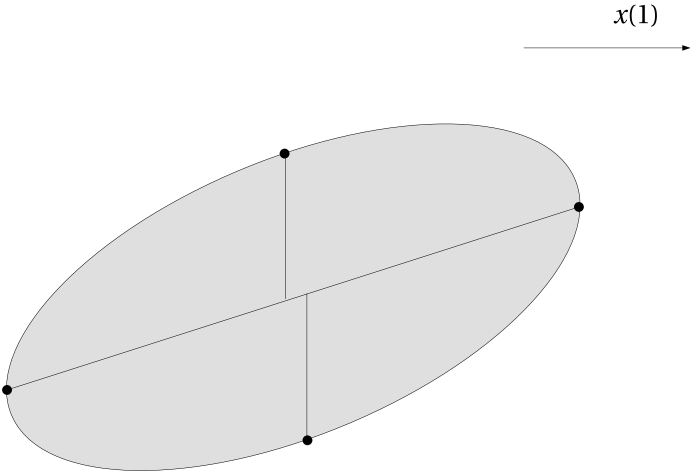
A linear program in the plane with m constraints can be solved in $O(m)$.

Combining Prune&Search with feasibility algorithm

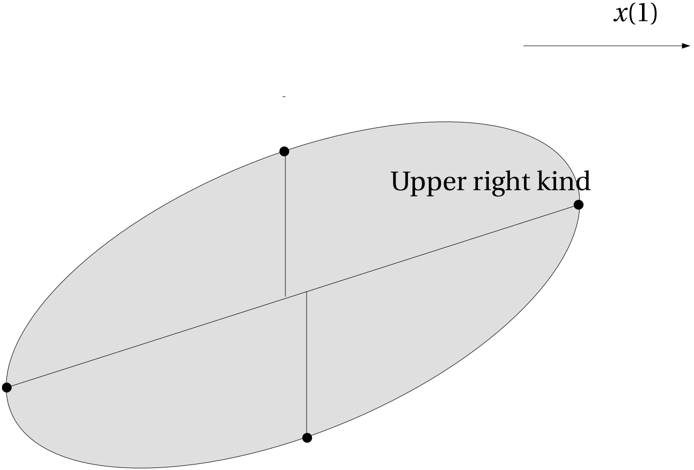
Partitioning the Polygon



Partitioning the Polygon

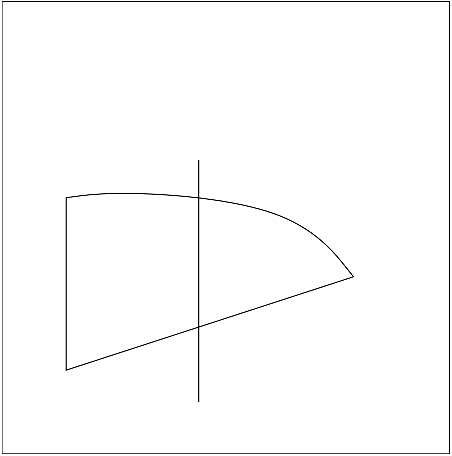


Partitioning the Polygon



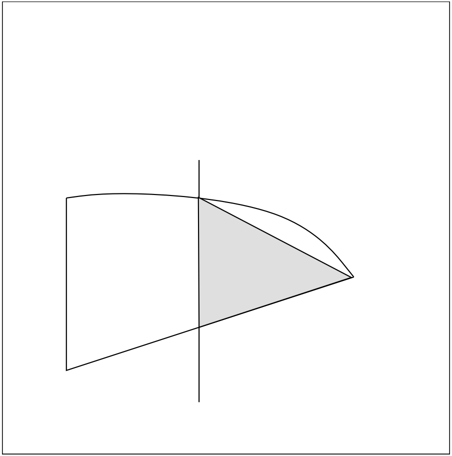
Upper right kind

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



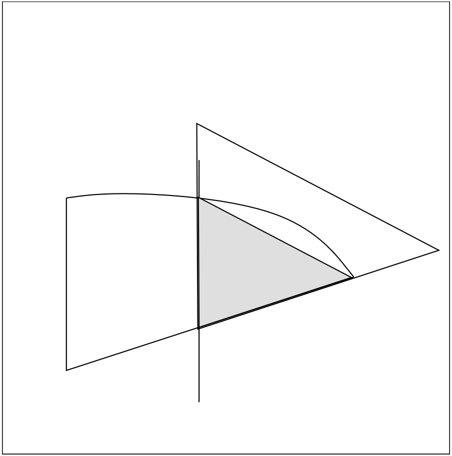
Upper right kind

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



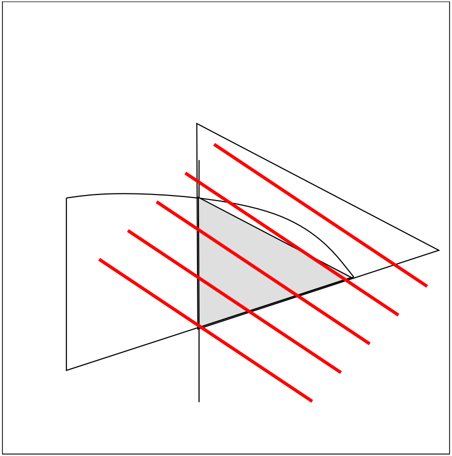
Upper right kind

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

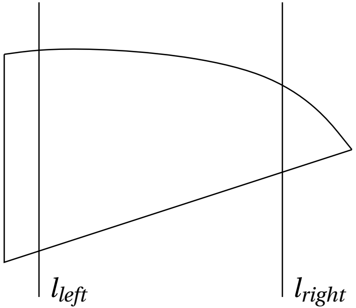


Upper right kind

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

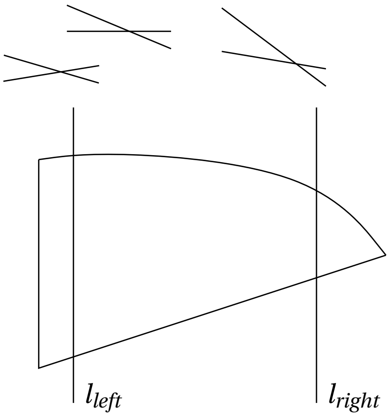


Prune & Search



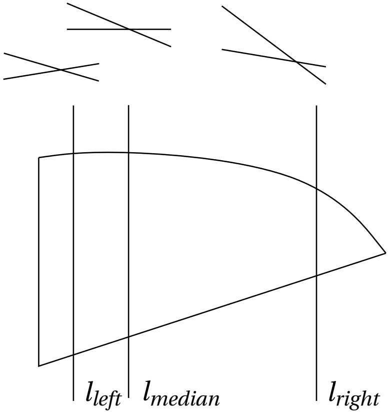
- Principle: Improve l_{left} and l_{right}

Prune & Search



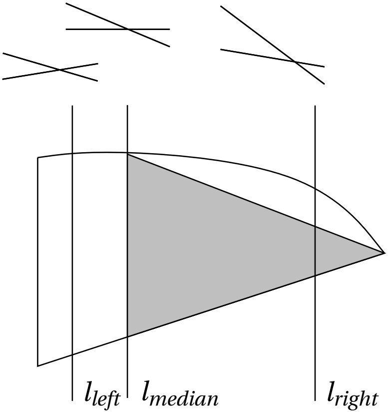
- Principle: Improve l_{left} and l_{right}
- Pair constraints arbitrarily

Prune & Search



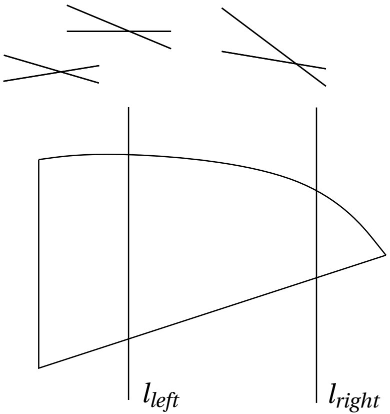
- Principle: Improve l_{left} and l_{right}
- Pair constraints arbitrarily
- Compute median of intersections

Prune & Search



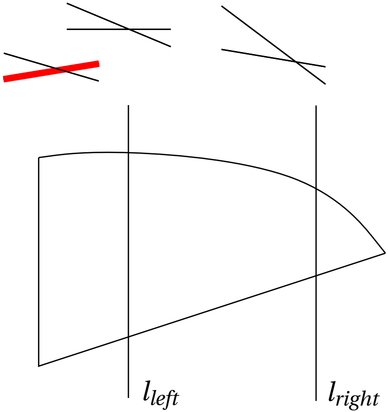
- Principle: Improve l_{left} and l_{right}
- Pair constraints arbitrarily
- Compute median of intersections
- Compute width of triangle defined by median

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: $O(\log m)$

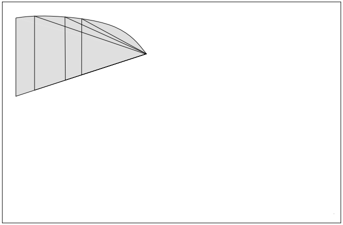
Analysis

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

Analysis

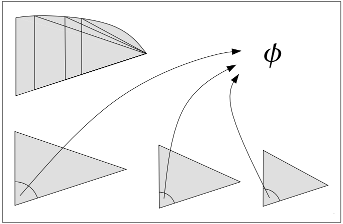
- Each round 1/4-th of constraints pruned
- Computing median is linear
- Running time without width checking: $O(m)$
- Number of checked triangles: $O(\log m)$
- Cost for width checking: $O(s)$
- Total cost $O(m + s \log m)$

The checked triangles



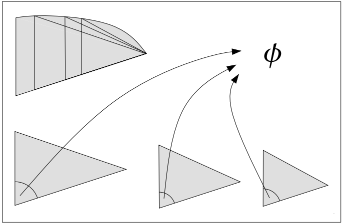
- Let $\begin{pmatrix} u^T \\ v^T \end{pmatrix}$ be the matrix of one prototype triangle

The checked triangles



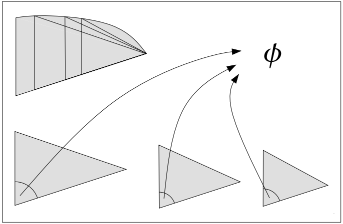
- Let $\binom{u^T}{v^T}$ be the matrix of one prototype triangle
- Query: Given $\alpha \in \mathbb{Q}$, $\beta \in \mathbb{Q}$ compute shortest vector of lattice $\Lambda = \{\alpha \binom{u^T}{\beta v^T} x \mid x \in \mathbb{Z}^2\}$

The checked triangles



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

The checked triangles



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

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

$$\Lambda = \{\binom{u^T}{\beta v^T} x \mid x \in \mathbb{Z}^2\}$$

Batching the width checks

Theorem

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

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

Batching the width checks

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

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

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

- 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: $\Lambda = \{\begin{pmatrix} a & b \\ 0 & \beta\, c \end{pmatrix} x \mid x \in \mathbb{Z}^2\}$

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

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

- 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 $O(m+s)$.

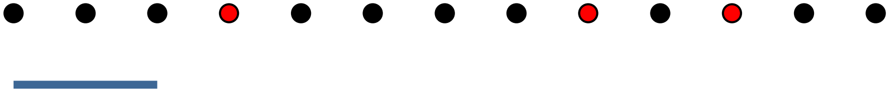
Linear Programming

Quiz

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

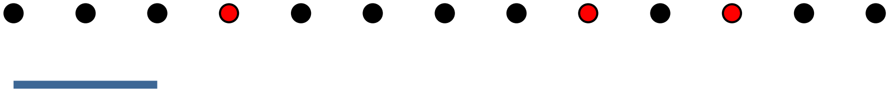
Quiz

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



Quiz

- $H = \{1, \ldots, n\}$, $r \in H$, $R \in \binom{H}{r}$ 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 $H^- = \{x(i) \leqslant M \mid i = 1, \dots, d\}$ explicit upper bounds
- For $G \subseteq H$, $x^*(G)$ is lex. max. point satisfying all $h \in G \cup H^-$
- Task: Compute $x^*(H)$

$B \subseteq H$ is called Basis of G, if $x^*(B) = x^*(H)$ and for each $b \in B$ one has $x^*(B - h) > x^*(B)$.

Lemma

Let B be a basis of H and let $G \subseteq H$. One has $x^*(G) > x^*(H)$ if and only if there exists $b \in B$ with $x^*(G)$ violates b.

Quiz

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

Quiz

- Choose $R \in \binom{H}{r}$ uniformly at random
- $V_R = \{h \in H \mid x^*(R)$ 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 $\chi(Q,h) = \begin{cases} 1 & \text{if } x^*(Q) \text{ violates } h, \\ 0 & \text{otherwise.} \end{cases}$

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

Sampling Lemma

Lemma (Clarlskon 1995 see also Gärtner & Welzl 1996)

Let G and H (multi-)sets of constraints $|H| = m$ and let $1 \leqslant r \leqslant m$. Then for random $R \in \binom{H}{r}$:

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

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

Sampling Lemma

Lemma (Clarlskon 1995 see also Gärtner & Welzl 1996)

Let G and H (multi-)sets of constraints $|H| = m$ and let $1 \leqslant r \leqslant m$. Then for random $R \in \binom{H}{r}$:

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

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

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

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

Clarkson's algorithm I

1. Input: H with $|H| = m$
2. $r \leftarrow d \cdot \sqrt{m}$

Clarkson's algorithm I

1. Input: H with $|H| = m$
2. $r \leftarrow d \cdot \sqrt{m}$ Sample size
3. $G \leftarrow \emptyset$

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`
   4.1 Choose random $R \in \binom{H}{r}$
   4.2 Compute $x^* = x^*(G \cup R)$

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`
   - 4.1 Choose random $R \in \binom{H}{r}$
   - 4.2 Compute $x^* = x^*(G \cup R)$ — with some other algorithm
   - 4.3 $V_R \leftarrow \{h \in H \mid x^* \text{ violates } h\}$
   - 4.4 `IF` $|V_R| \leqslant 2\sqrt{m}$

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`
   4.1 Choose random $R \in \binom{H}{r}$
   4.2 Compute $x^* = x^*(G \cup R)$ — with some other algorithm
   4.3 $V_R \leftarrow \{h \in H \mid x^* \text{ violates } h\}$
   4.4 `IF` $|V_R| \leqslant 2\sqrt{m}$ — With probability $\geqslant 1/2$ true
   `THEN` $G \leftarrow G \cup V_R$,

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`
   - 4.1 Choose random $R \in \binom{H}{r}$
   - 4.2 Compute $x^* = x^*(G \cup R)$ — with some other algorithm
   - 4.3 $V_R \leftarrow \{h \in H \mid x^* \text{ violates } h\}$
   - 4.4 `IF` $|V_R| \leqslant 2\sqrt{m}$ — With probability $\geqslant 1/2$ true
     `THEN` $G \leftarrow G \cup V_R$, — successful iteration
5. `UNTIL` $V_R = \emptyset$

Clarkson's algorithm I

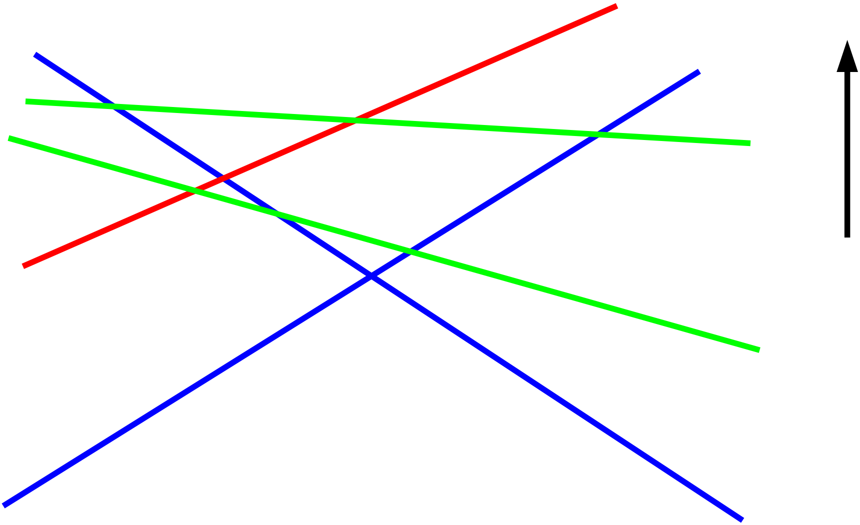
1. Input: H with $|H| = m$
2. $r \leftarrow d \cdot \sqrt{m}$ — Sample size
3. $G \leftarrow \emptyset$ — Contains optimal basis in the end
4. `REPEAT`
   - 4.1 Choose random $R \in \binom{H}{r}$
   - 4.2 Compute $x^* = x^*(G \cup R)$ — with some other algorithm
   - 4.3 $V_R \leftarrow \{h \in H \mid x^* \text{ violates } h\}$
   - 4.4 `IF` $|V_R| \leqslant 2\sqrt{m}$ — With probability $\geqslant 1/2$ 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

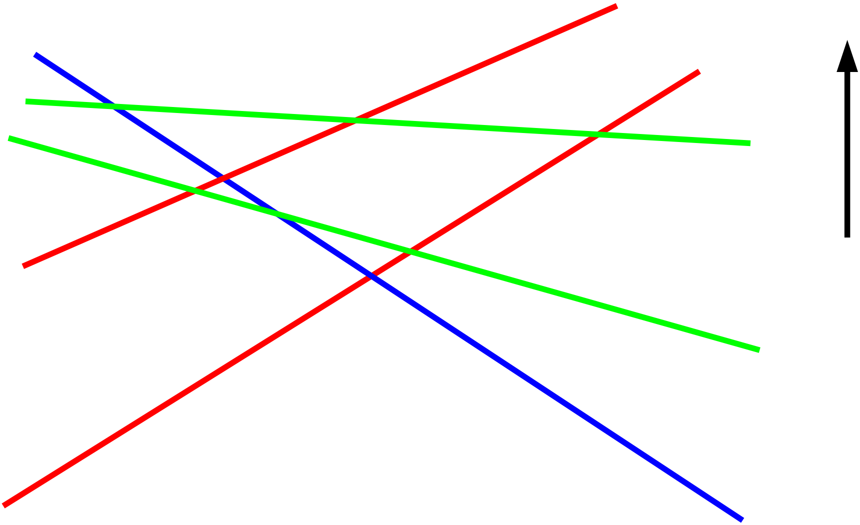
Example

R, G, B



Example

R, G, B



Analysis

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

Let B be optimal basis.

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

Clarkson 1 performs:

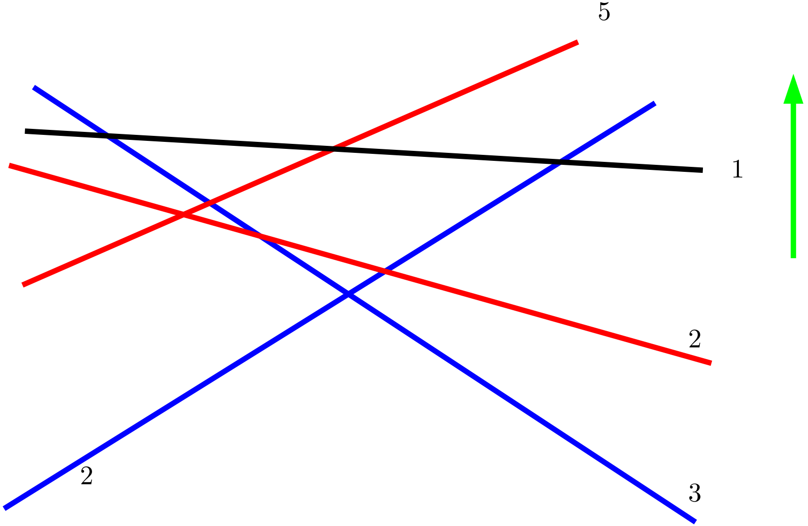
- Expected $2\,d$ calls to linear programming oracle with at most $3 \cdot d\sqrt{m}$ constraints
- Expected number of $O(d^2 \cdot m)$ arithmetic operations

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

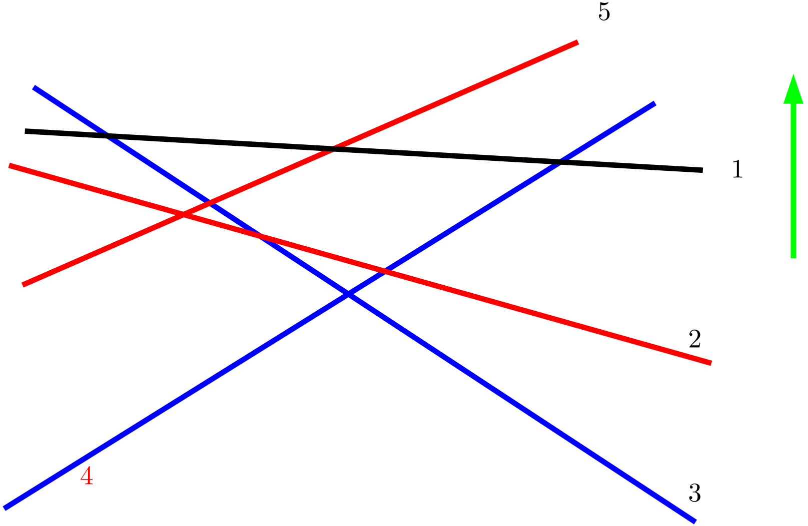
Example

R, B



Example

R, B



Clarkson 2

1. `INPUT:` H, $|H| = m$
2. $r \leftarrow 6 \cdot d^2$

Clarkson 2

1. `INPUT:` H, $|H| = m$
2. $r \leftarrow 6 \cdot d^2$ sample size
3. `REPEAT:`
   3.1 Choose random $R \in \binom{H}{r}$

Clarkson 2

1. `INPUT:` H, $|H| = m$
2. $r \leftarrow 6 \cdot d^2$ — sample size
3. `REPEAT:`
   3.1 Choose random $R \in \binom{H}{r}$ — will contain optimum basis
   3.2 Compute $x^* = x^*(R)$,

Clarkson 2

1. `INPUT:` H, $|H| = m$
2. $r \leftarrow 6 \cdot d^2$ — sample size
3. `REPEAT:`
   - 3.1 Choose random $R \in \binom{H}{r}$ — will contain optimum basis
   - 3.2 Compute $x^* = x^*(R)$, — with some other algorithm
   - 3.3 $V_R \leftarrow \{h \in H \mid x^* \text{ violates } h\}$
   - 3.4 `IF` $\mu(V_R) \leqslant 1/(3d)\mu(H)$

Clarkson 2

1. `INPUT`: H, $|H| = m$
2. $r \leftarrow 6 \cdot d^2$ — sample size
3. `REPEAT`:
   - 3.1 Choose random $R \in \binom{H}{r}$ — will contain optimum basis
   - 3.2 Compute $x^* = x^*(R)$, — with some other algorithm
   - 3.3 $V_R \leftarrow \{h \in H \mid x^* \text{ violates } h\}$
   - 3.4 `IF` $\mu(V_R) \leqslant 1/(3d)\mu(H)$ — probability $\geqslant 1/2$

     `THEN` for all $h \in V$ do $\mu_h \leftarrow 2\mu_h$

Clarkson 2

1. `INPUT:` H, $|H| = m$
2. $r \leftarrow 6 \cdot d^2$ — sample size
3. `REPEAT:`
   - 3.1 Choose random $R \in \binom{H}{r}$ — will contain optimum basis
   - 3.2 Compute $x^* = x^*(R)$, — with some other algorithm
   - 3.3 $V_R \leftarrow \{h \in H \mid x^* \text{ violates } h\}$
   - 3.4 `IF` $\mu(V_R) \leqslant 1/(3d)\mu(H)$ — probability $\geqslant 1/2$
     `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):

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

Proof:

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

Complexity Clarkson 2

- $2^k \leqslant me^{k/3}$ 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

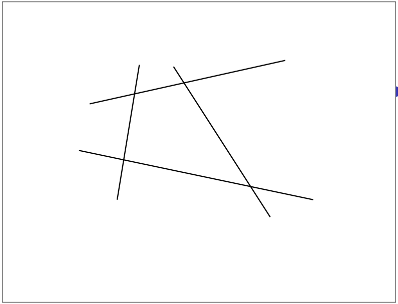
- Given set H of m integral constraints in dimension d and $H^- = \{x(i) \leqslant M \mid i = 1, \ldots, d\}$ explicit bound constraints.
- For $G \subseteq H$, $x^*(G)$ is lex. max. integer point satisfying G and H^-.
- Task: Compute $x^*(H)$.

A theorem of Bell and Scarf

Theorem

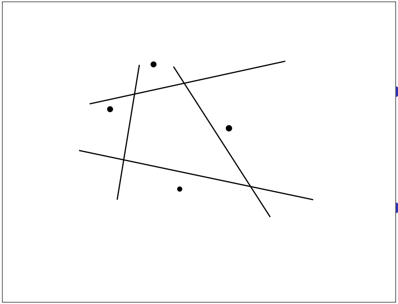
Let H be a set of rational linear constraints in $\mathbb{R}^d$. If there does not exist an integer point which satisfies all constraints, then there exists a subset $B \subseteq H$ with $|B| \leqslant 2^d$ such that there does not exist an integer point which satisfies all constraints in B.

Proof



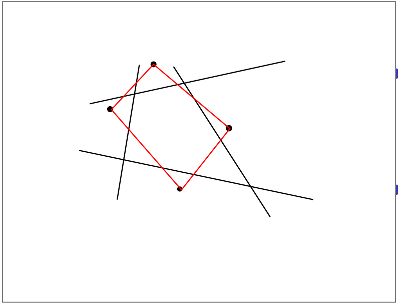
- Let H be minimal such that H has no feasible integer point, $m = |H| > 2^d$
- Assume constraints are $a_i^T x \leqslant \beta_i$ $i = 1, .., m$, where a_i and β_i are integers

Proof



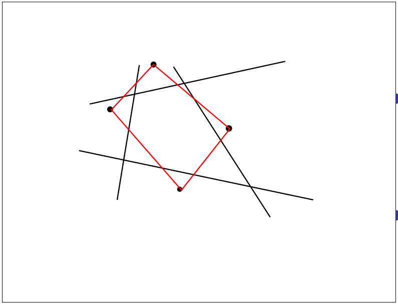
- Let H be minimal such that H has no feasible integer point, $m = |H| > 2^d$
- Assume constraints are $a_i^T x \leqslant \beta_i$ $i = 1, .., m$, where a_i and β_i are integers
- For each $a_i^T x \leqslant \beta_i$, there exists integer solution y_i which satisfies all but the i-th constraint.

Proof



- Let H be minimal such that H has no feasible integer point, $m = |H| > 2^d$
- Assume constraints are $a_i^T x \leqslant \beta_i$ $i = 1, .., m$, where a_i and β_i are integers
- For each $a_i^T x \leqslant \beta_i$, 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 $\gamma_1, \ldots, \gamma_m \in \mathbb{Z}$ s.t. $\beta_i \leqslant \gamma_i$, system $a_i^T x \leqslant \gamma_i, i = 1, \ldots, m$ has no solution in Z and $\gamma_1 + \cdots + \gamma_m$ is maximal
- For each i there exists a $z_i \in Z$ s.t. $a_i^T z_i = \gamma_i + 1$ and $a_j^T z_i \leqslant \gamma_j$ 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 $|B| \leqslant 2^d - 1$ 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.
- IP with fixed number of constraints can be solved in time $O(s)$
- Total running time: Expected $O(m + \log m + s)$