Announcement:
Thursday, May 20th, we will have the lecture at the usual time (10:15–12) on Zoom, inspite of the public holiday in Germany.

The lecture will be recorded. If you want to observe the holiday you can still view the lecture.

Linear Programming for few variables and many constraints

Given finite set H of size $n > d$ and
for each $h \in H$ some $a_h \in \mathbb{R}^d \setminus \{0\}$
and some $b_h \in \mathbb{R}$
and given some $c \in \mathbb{R}^d \setminus \{0\}$

find some $x \in \mathbb{R}^d$ so as to

minimize $\langle c, x \rangle$
s.t. $\langle a_h, x \rangle \leq b_h$ for each $h \in H$.

Linear Programming for few variables and many constraints

Given finite set H of size $n > d$ and
for each $h \in H$ some $a_h \in \mathbb{R}^d \setminus \{0\}$
and some $b_h \in \mathbb{R}$
and given some $c \in \mathbb{R}^d \setminus \{0\}$

find some $x \in \mathbb{R}^d$ so as to

minimize $\langle c, x \rangle$
s.t. $\langle a_h, x \rangle \leq b_h$ for each $h \in H$.

$$\langle u, v \rangle = \sum_{1 \leq i \leq d} u_i v_i$$

Linear Programming for few variables and many constraints

Given finite set H of size $n > d$ and
for each $h \in H$ some $a_h \in \mathbb{R}^d \setminus \{0\}$
and some $b_h \in \mathbb{R}$
and given some $c \in \mathbb{R}^d \setminus \{0\}$

find some $x \in \mathbb{R}^d$ so as to

minimize $\langle c, x \rangle$
s.t. $\langle a_h, x \rangle \leq b_h$ for each $h \in H$.

Given finite set H of n halfspaces in $\mathbb{R}^d$
and some direction c

find some $x \in \bigcap H$
that is least in direction c

Linear Programming for few variables and many constraints

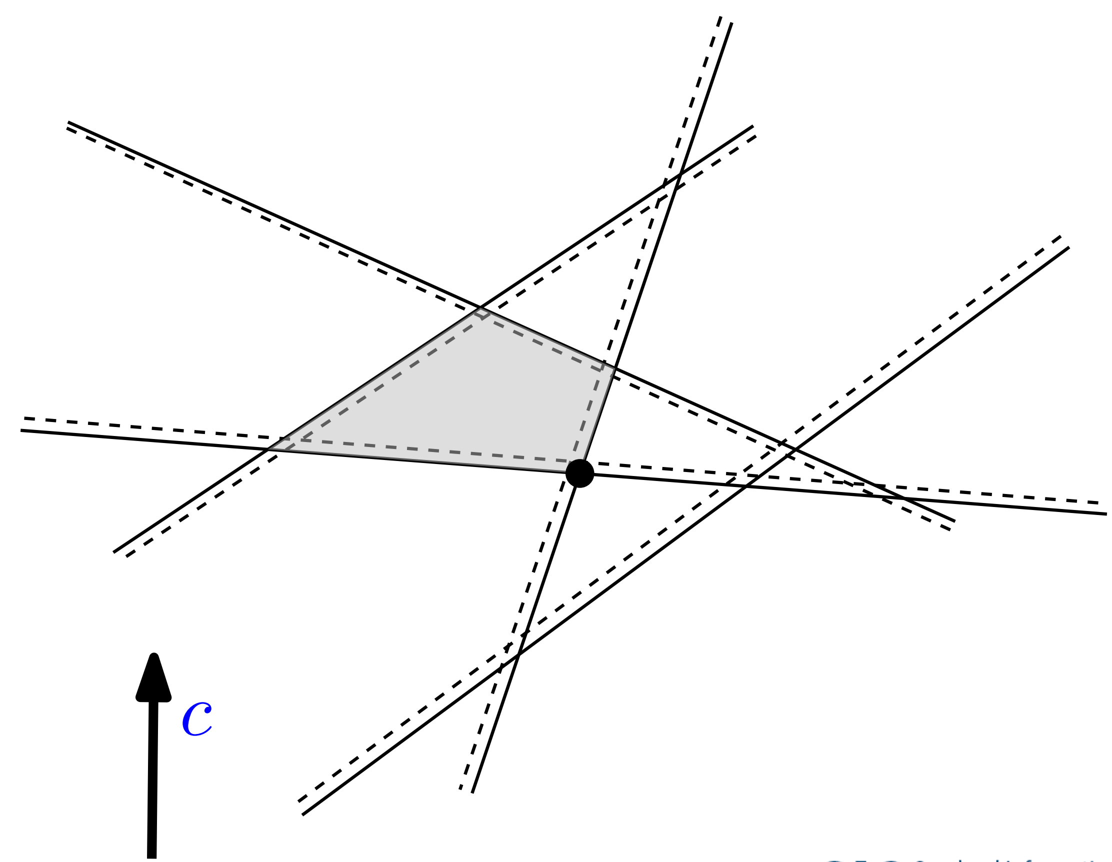
Given finite set H of size $n > d$ and
for each $h \in H$ some $a_h \in \mathbb{R}^d \setminus \{0\}$
and some $b_h \in \mathbb{R}$
and given some $c \in \mathbb{R}^d \setminus \{0\}$

find some $x \in \mathbb{R}^d$ so as to

minimize $\langle c, x \rangle$
s.t. $\langle a_h, x \rangle \leq b_h$ for each $h \in H$.

Given finite set H of n halfspaces in $\mathbb{R}^d$
and some direction c

find some $x \in \bigcap H$
that is least in direction c



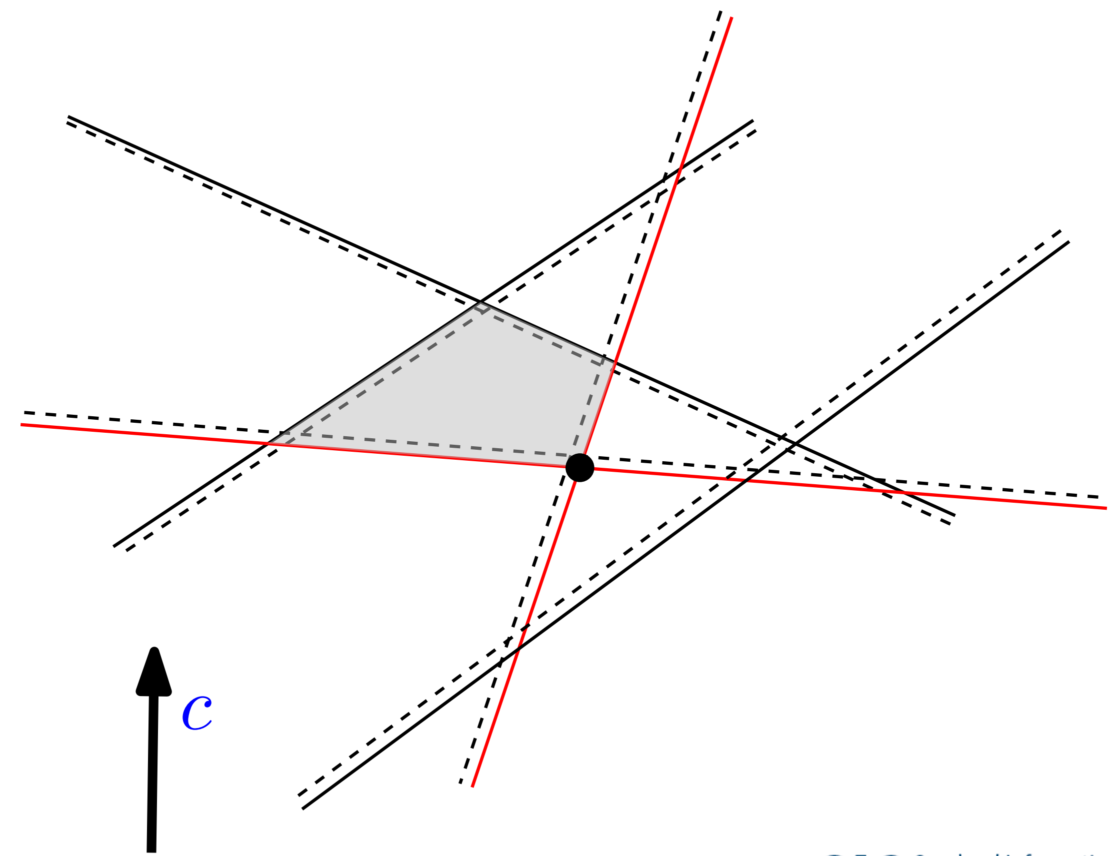
Linear Programming for few variables and many constraints

Given finite set H of n halfspaces in $\mathbb{R}^d$
and some direction c

find some $x \in \bigcap H$
that is least in direction c

Possibility 1: An optimal solution exisits.

Given by an intersection point of d bounding hyperplanes from H



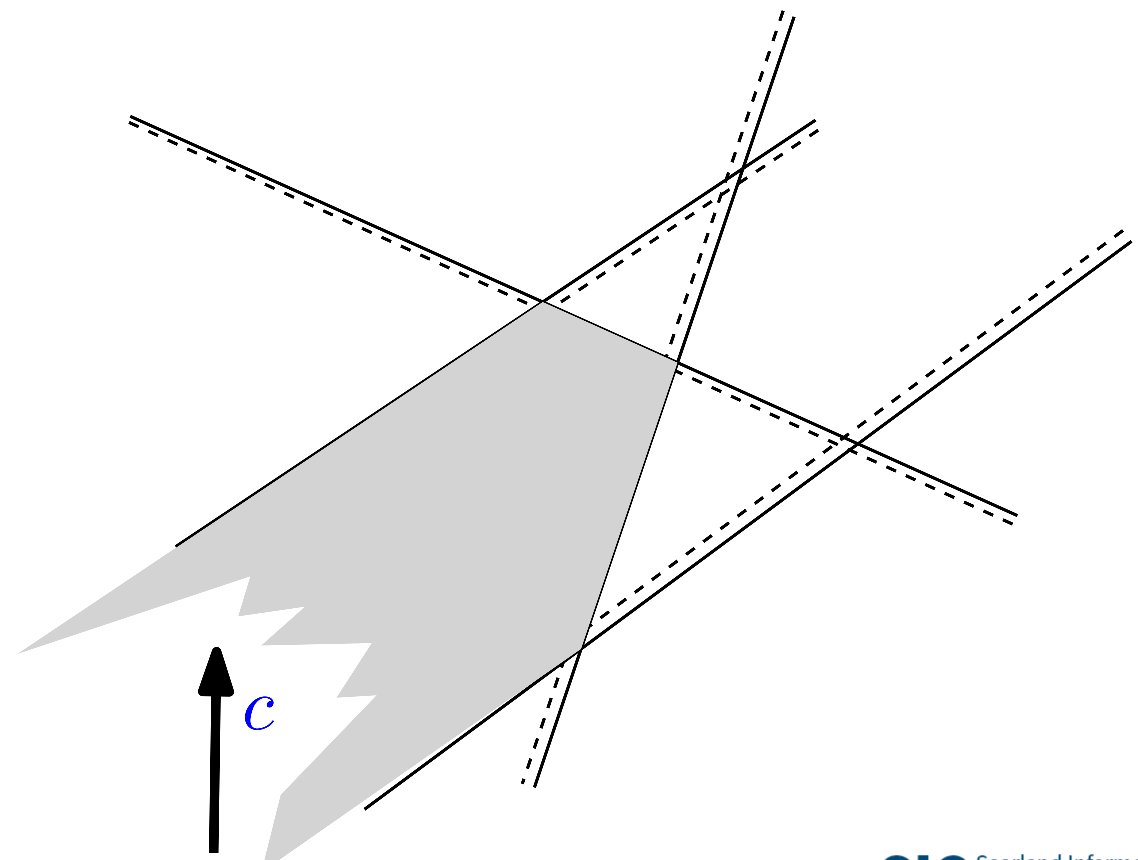
Linear Programming for few variables and many constraints

Possibility 2: No optimal solution exisits.

The intersection is unbounded.

Given finite set H of n halfspaces in $\mathbb{R}^d$ and some direction c

find some $x \in \bigcap H$ that is least in direction c



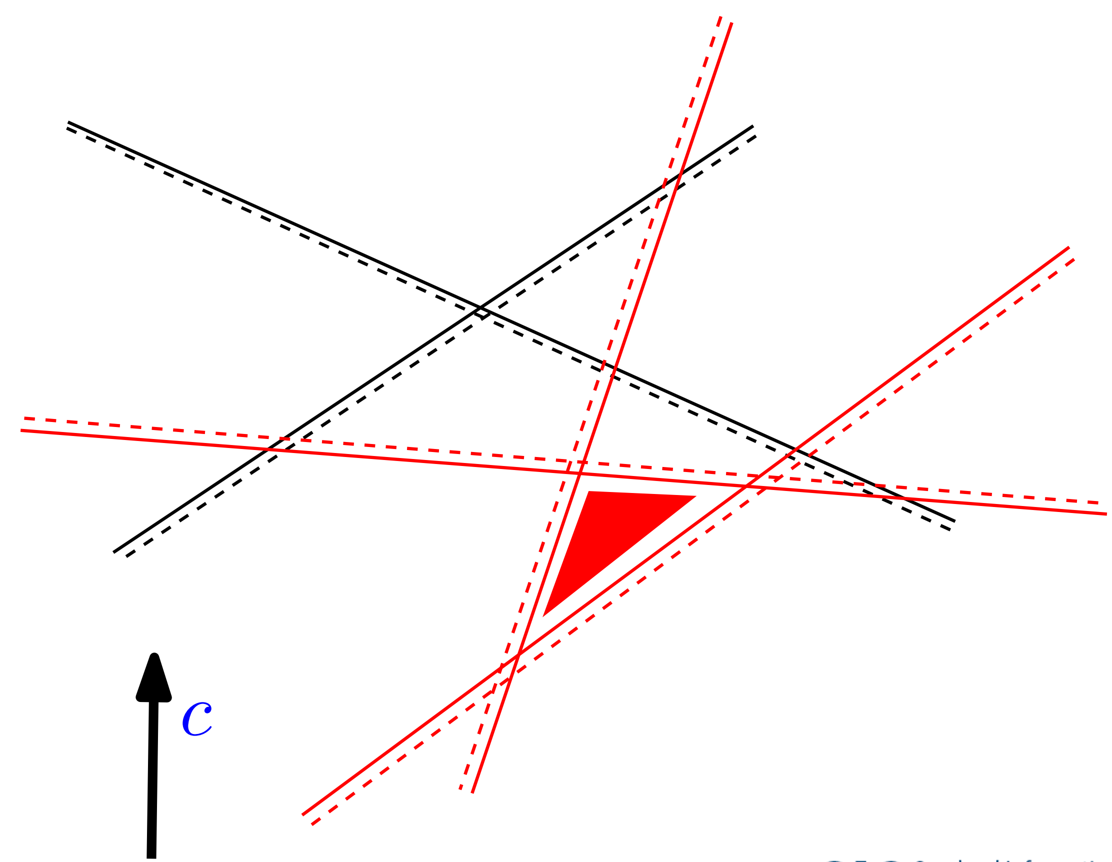
Linear Programming for few variables and many constraints

Given finite set H of n halfspaces in $\mathbb{R}^d$
and some direction c

find some $x \in \bigcap H$
that is least in direction c

Possibility 3: No solution exisits because the problem is infeasible, i.e. $\bigcap H$ is empty.

In this case there must be $d+1$ halfspaces in H that do not intersect.



LP — Known Methods and Results

Fourier–Motzkin elimination	slow
Simplex method	fast in practice, bad in the worst case
Ellipsoid method	good (polynomial time) theoretical bounds
Interior point methods	good theoretical bounds, can be made fast in practice
...	

Polynomial time methods work in the bit model of computation and need inputs to be integral.

No polynomial time algorithm is know for the algebraic model, where you count operations on numbers (and not on bits).

LP for $d = 2$ variables and many constraints

Mainly intersted in algorithmic aspects.

Simplifying assumptions:

- optimization direction is vertical
- only upper halfplanes
- no degeneracies

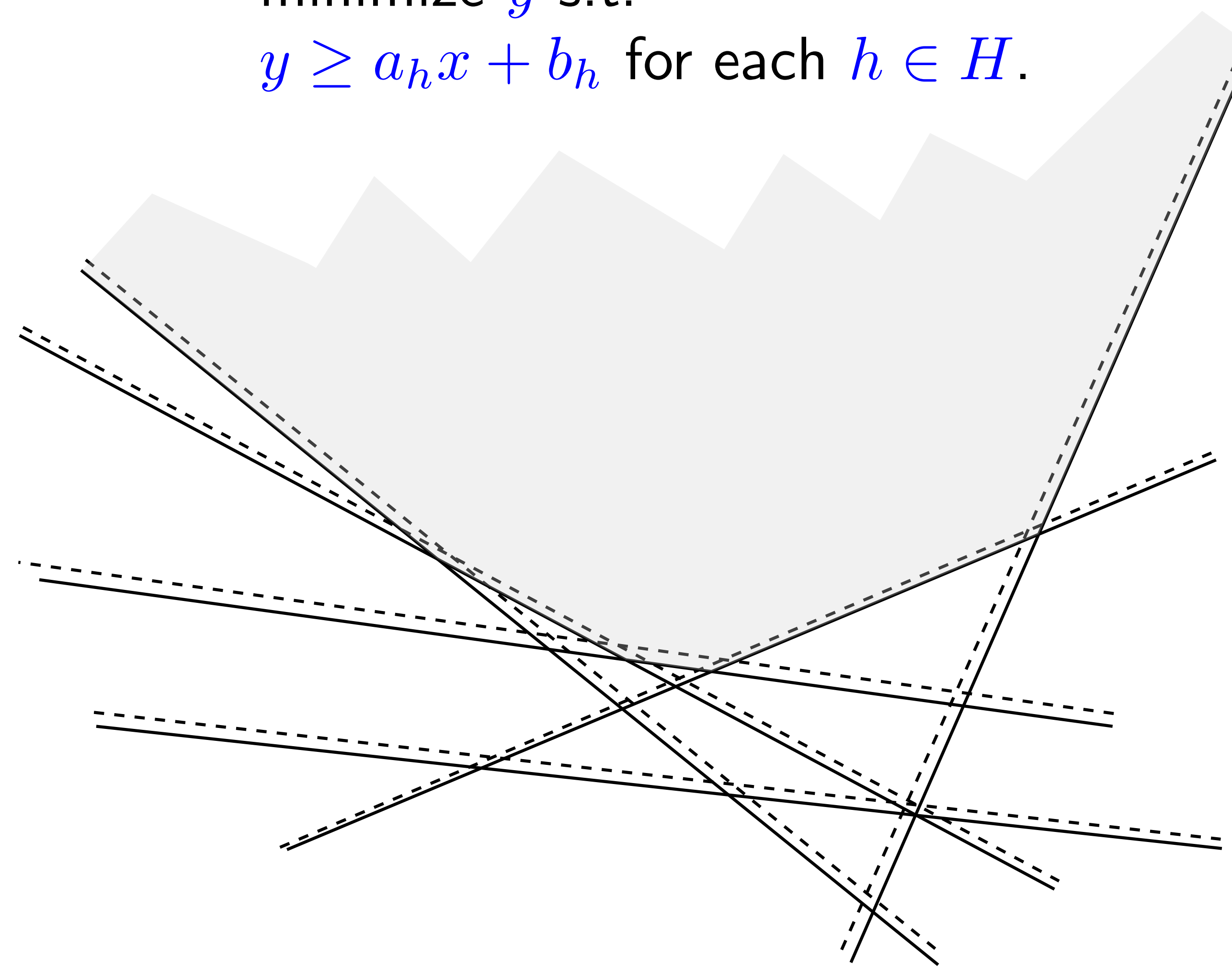
LP for $d = 2$ variables and many constraints

Mainly intersted in algorithmic aspects.

Simplifying assumptions:

- optimization direction is vertical
- only upper halfplanes
- no degeneracies

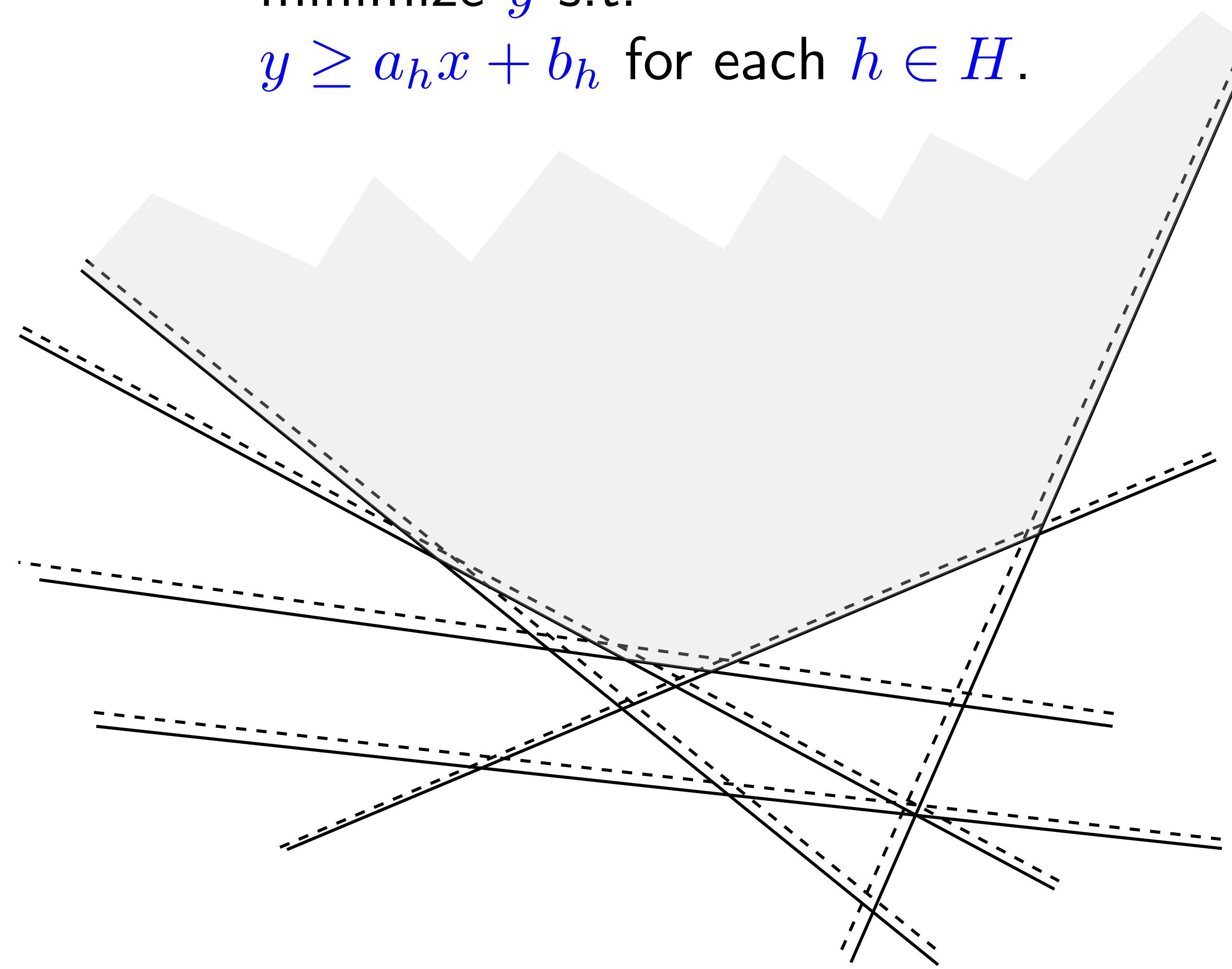
minimize y s.t.
$y \geq a_h x + b_h$ for each $h \in H$.



LP for $d = 2$ variables and many constraints

Mainly intersted in algorithmic aspects.

minimize y s.t.
$y \geq a_h x + b_h$ for each $h \in H$.



Easy Algorithm:

1. Compute the boundary of the intersection of the upper halfplanes.

2. Find "lowest" point on that boundary

$O(n \log n)$ time

LP for $d = 2$ variables and many constraints

Mainly intersted in algorithmic aspects.

Decimation Algorithm:
Megiddo 1982, Dyer 1982

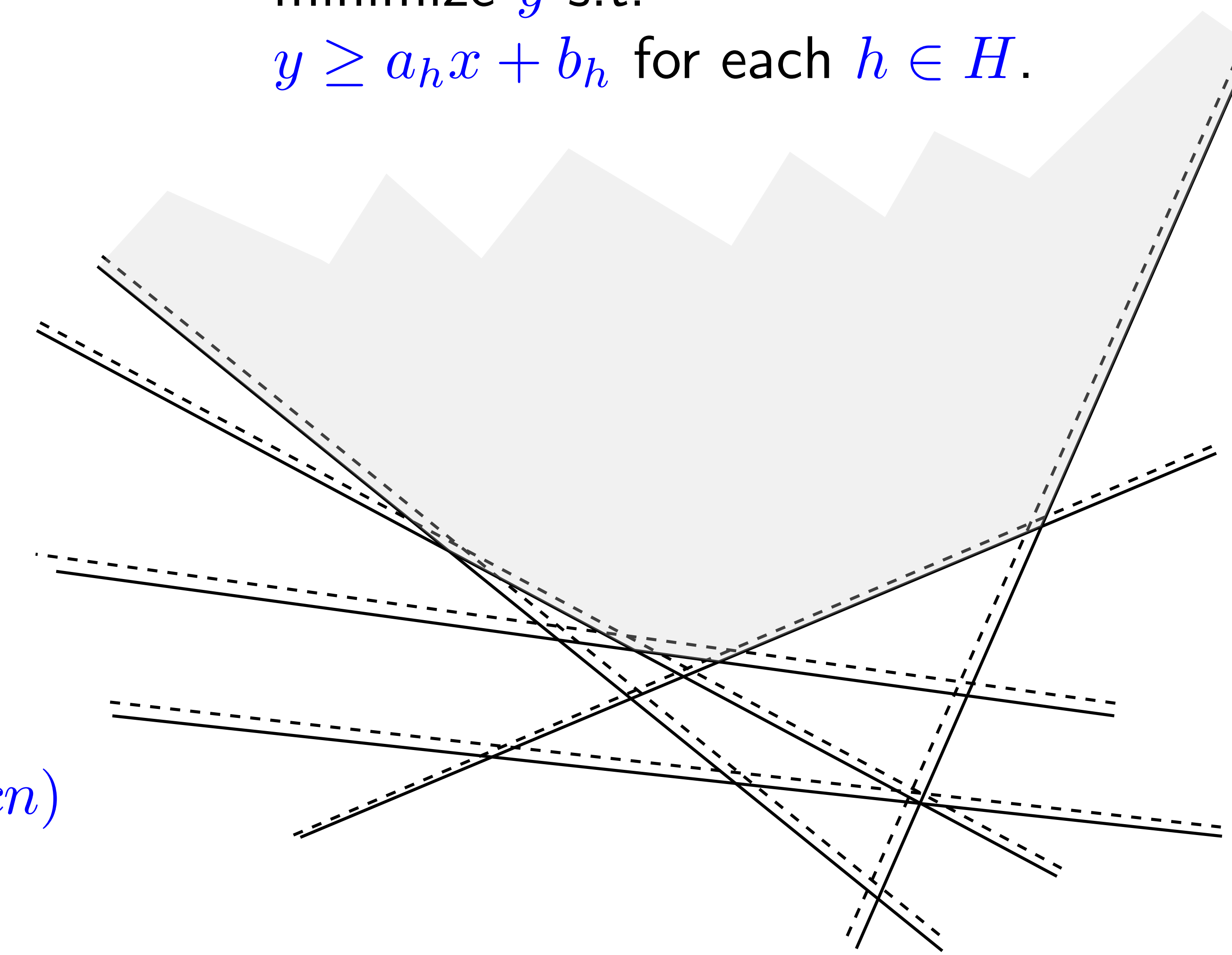
(i) Identify in linear time a constant fraction of the halfplanes that cannot possibly contribute to the optimum point.

(ii) Remove those halfplanes from consideration and recurse.

Running Time: $T(n) \leq O(n) + T(cn)$
for some $c < 1$.

$\Longrightarrow \quad T(n) = O(n)$

minimize y s.t.
$y \geq a_h x + b_h$ for each $h \in H$.



SIC Saarland Informatics Campus

How to eliminate

How to eliminate

Removing Simplifying Assumptions

Simplifying assumptions:

- optimization direction is vertical
- only upper halfplanes
- no degeneracies

LP for $d = 3$ variables and many constraints

Simplifying assumptions:

- optimization direction is vertical
- only upper halfspaces
- no degeneracies

minimize z s.t.
$z \geq \alpha_h x + \beta_h y + b_h$ for each $h \in H$.

Easy Algorithm:

1. Compute the boundary of the intersection of the upper halfplanes.
2. Find "lowest" point on that boundary

$O(n \log n)$ time (the first step can be done by 3d convex hull algorithm)

LP for $d = 3$ variables and many constraints

Decimation Algorithm:

Megiddo 1982, Dyer 1982

(i) Identify in linear time a constant fraction of the halfspaces that cannot possibly contribute to the optimum point.

(ii) Remove those halfspaces from consideration and recurse.

LP for $d = 3$ variables and many constraints

Decimation Algorithm:

Megiddo 1982, Dyer 1982

(i) Identify in linear time a constant fraction of the halfspaces that cannot possibly contribute to the optimum point. "redundant halfspaces"

(ii) Remove those redundant halfspaces from consideration and recurse.

How to identify a redundant halfspace

How to do an optimum-location query

How to answer many optimum-location queries using just two actual queries.

Abstract Problem: Given a set L of m lines in the plane and an *oracle* that tells, on which side of a query line lies an (unknown) target point, decide for many lines in L which side contains the target point.

Claim: With just two queries to the oracle for $m/4$ of the lines in L the location of the target point can be decided.

Corollary: Given a set of n (upper) halfspaces in $\mathbb{R}^3$ in linear time $n/8$ redundant halfspaces can be identified.

Theorem: The lowest point in the intersection of n (upper) halfspaces in $\mathbb{R}^3$ can be found in linear time.

“Linear Programming with 3 variables can be solved in linear time.”

Removing Simplifying assumptions: (exercise!)

- optimization direction is vertical
- only upper halfspaces
- no degeneracies

LP in constant dimension

Theorem: Megiddo (1984)
For any fixed d a linear program in d variables and with n constraints can be solved in $O(n)$ time.

LP in constant dimension

Theorem: Megiddo (1984)
For any fixed d a linear program in d variables and with n constraints can be solved in $O(2^{2^d} n)$ time.

A simple randomized LP algorithm

A simple randomized LP algorithm

function SLP(H : set of halfspaces in $\mathbb{R}^d$, c : direction in $\mathbb{R}^d$) : optimum point

if $|H| \leq d$ **then** solve by brute force

else choose some $h \in H$ uniformly at random

$x =$SLP($H \setminus \{h\}$, c)

if $x \in h$ **then** return x

else let $H' = \{f \cap h^\circ \mid f \in H \setminus \{h\}\}$

let c' be projection of c into h°

return SLP(H', c')

A simple randomized LP algorithm

function SLP(H : set of halfspaces in $\mathbb{R}^d$, c : direction in $\mathbb{R}^d$) : optimum point

if $|H| \le d$ **then** solve by brute force
else choose some $h \in H$ uniformly at random
$x =$SLP($H \setminus \{h\}$, c)
if $x \in h$ **then** return x
else let $H' = \{f \cap h^\circ \mid f \in H \setminus \{h\}\}$
let c' be projection of c into h°
return SLP(H', c')

Claim; If h is chosen uniformly at random from the n halfspaces in H then the LP-Optimum for H differs from the LP-Optimum for $H \setminus \{h\}$ with probability at most d/n.

A simple randomized LP algorithm

function SLP(H : set of halfspaces in $\mathbb{R}^d$, c : direction in $\mathbb{R}^d$) : optimum point

if $|H| \leq d$ **then** solve by brute force
else choose some $h \in H$ uniformly at random
$x =$SLP($H \setminus \{h\}$, c)
if $x \in h$ **then** return x
else let $H' = \{f \cap h^\circ \mid f \in H \setminus \{h\}\}$
let c' be projection of c into h°
return SLP(H', c')

Claim; If h is chosen uniformly at random from the n halfspaces in H then the LP-Optimum for H differs from the LP-Optimum for $H \setminus \{h\}$ with probability at most d/n.

Expected running time $T(n,d) \leq T(n-1,d) + \frac{d}{n}\left(\alpha dn + T(n-1,d-1)\right)$.

SIC Saarland Informatics Campus

A simple randomized LP algorithm

function SLP(H : set of halfspaces in $\mathbb{R}^d$, c : direction in $\mathbb{R}^d$) : optimum point

if $|H| \leq d$ **then** solve by brute force
else choose some $h \in H$ uniformly at random
$x =$SLP($H \setminus \{h\}$, c)
if $x \in h$ **then** return x
else let $H' = \{f \cap h^\circ \,|\, f \in H \setminus \{h\}\}$
let c' be projection of c into h°
return SLP(H', c')

Claim; If h is chosen uniformly at random from the n halfspaces in H then the LP-Optimum for H differs from the LP-Optimum for $H \setminus \{h\}$ with probability at most d/n.

Expected running time $T(n,d) \leq T(n-1,d) + \frac{d}{n}\,(\alpha dn + T(n-1,d-1))$.

$$\Longrightarrow \qquad T(n,d) = O(d!\,n)$$

A sampling LP algorithm

H set of n halfspaces in $\mathbb{R}^d$;
we want to find the "lowest" point in their intersection

Assume you have some alternative LP algorithm ALP() for "small" input sets available

A sampling LP algorithm

H set of n halfspaces in $\mathbb{R}^d$;
we want to find the "lowest" point in their intersection

Assume you have some alternative LP algorithm ALP() for "small" input sets available

function Sample-LP(H) : optimum point
if $n \leq 9d^2$ **then** return ALP(H)
else
  $r := d\sqrt{n}$; G:={};
  repeat
    choose random $R \in \binom{H}{r}$
    v :=ALP($G \cup R$)
    $V := \{h \in H | v$ "violates" h, i.e. $v \notin h\}$
    if $|V| \leq 2\sqrt{n}$ **then** $G := G \cup V)$
  until $V = \emptyset$
  return v

A sampling LP algorithm

H set of n halfspaces in $\mathbb{R}^d$;
we want to find the "lowest" point in their intersection

Assume you have some alternative LP algorithm ALP() for "small" input sets available

function Sample-LP(H) : optimum point
if $n \leq 9d^2$ **then** return ALP(H)
else
  $r := d\sqrt{n}$; G:={};
  repeat
    choose random $R \in \binom{H}{r}$
    v :=ALP($G \cup R$)
    $V := \{h \in H | v$ "violates" h, i.e. $v \notin h\}$
    if $|V| \leq 2\sqrt{n}$ **then** $G := G \cup V)$
  until $V = \emptyset$
  return v

Claim:
- exp. number of calls to ALP() is $\leq 2d$
- in each such call the input contains $\leq 3d\sqrt{n}$ halfspaces
- exp. number of arithmetic operations (in violation tests) is $O(d^2 n)$

A sampling LP algorithm

H set of n halfspaces in $\mathbb{R}^d$;
we want to find the "lowest" point in their intersection

Assume you have some alternative LP algorithm ALP() for "small" input sets available

function Sample-LP(H) : optimum point
if $n \leq 9d^2$ **then** return ALP(H)
else
 $r := d\sqrt{n}$; G:={};
 repeat
  choose random $R \in \binom{H}{r}$
  v :=ALP($G \cup R$)
  $V := \{h \in H | v$ "violates" h, i.e. $v \notin h\}$
  if $|V| \leq 2\sqrt{n}$ **then** $G := G \cup V)$
 until $V = \emptyset$
 return v

Claim:

- exp. number of calls to ALP() is $\leq 2d$
- in each such call the input contains $\leq 3d\sqrt{n}$ halfspaces
- exp. number of arithmetic operations (in violation tests) is $O(d^2 n)$

Problem of size n is reduced to $\leq 2d$ problems of size $O(d\sqrt{n})$ in expected time $O(d^2 n)$.

A Sampling Lemma

Lemma: Let G and H be finite sets of halfspaces in $\mathbb{R}^d$, and let $1 \leq r \leq n = |H|$.

Let R be a random subset of H of size r.

Let $V_R = \{h \in H | h$ violates the optimum for $G \cup R\}$.

Then in expectation the size of V_R is at most $d\frac{n-r}{r+1}$.

Best current time bounds

Combination of two kinds of sampling algorithm, a non-trivial improvement of the simple incremental linear programmong algorithm (plus rather fancy generating function based analysis) yields algorithm with expected running time

$$O(d^2 n + e^{O(\sqrt{d \log d})})$$

Clarkson, Matousek, Sharir, Welzl, Gärtner, Kalai