Linear arithmetic theories:
(integer) linear programming

Christoph Haase   Alessio Mansutti

ESSLLI 2023

Today's lecture:
Linear programming (LP) and Integer Linear programming (ILP)

- Introduction to LP and ILP (geometry, basic properties)
- Algorithms for LP: Simplex algorithm, ellipsoid method
- Algorithm for ILP: Branch-and-bound
- Good solutions for ILP: Randomized rounding
- The computational complexity of linear and integer programming
- Tool demo along the way: SAGE and 4TI2

Introduction to (Integer) Linear Programming

Linear programming

Given:

- Polyhedron in $\mathbb{R}^d$ defined by conjunction of linear constraints:

$$A \cdot \boldsymbol{x} \geq \boldsymbol{c}$$

- Linear objective function $\boldsymbol{b}^\intercal \cdot \boldsymbol{x}$

Linear programming

Given:

- Polyhedron in $\mathbb{R}^d$ defined by conjunction of linear constraints:

$$A \cdot \boldsymbol{x} \geq \boldsymbol{c}$$

- Linear objective function $\boldsymbol{b}^\intercal \cdot \boldsymbol{x}$

Goal:

- Maximize objective function while respecting all constraints

Linear programming

Given:

- Polyhedron in $\mathbb{R}^d$ defined by conjunction of linear constraints:

$$A \cdot \boldsymbol{x} \geq \boldsymbol{c}$$

- Linear objective function $\boldsymbol{b}^\intercal \cdot \boldsymbol{x}$

Goal:

- Maximize objective function while respecting all constraints

Important aspects:

- Decide feasibility
- Decide boundedness of objective function
- Understand where to find optima

Integer Linear programming

Given:

- Polyhedron in $\mathbb{Z}^d$ defined by conjunction of linear constraints:

$$A \cdot \boldsymbol{x} \geq \boldsymbol{c}$$

- Linear objective function $\boldsymbol{b}^\intercal \cdot \boldsymbol{x}$

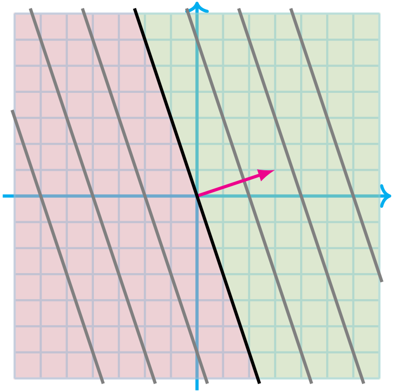
Goal:

- Maximize objective function while respecting all constraints

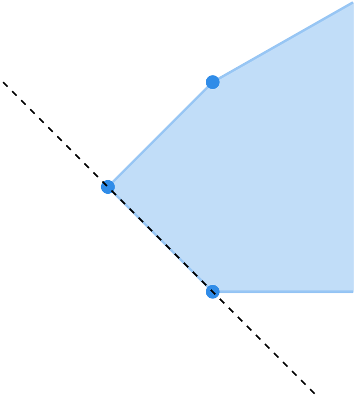
Important aspects:

- Decide feasibility
- Decide boundedness of objective function
- Understand where to find optima

Yesterday we saw...

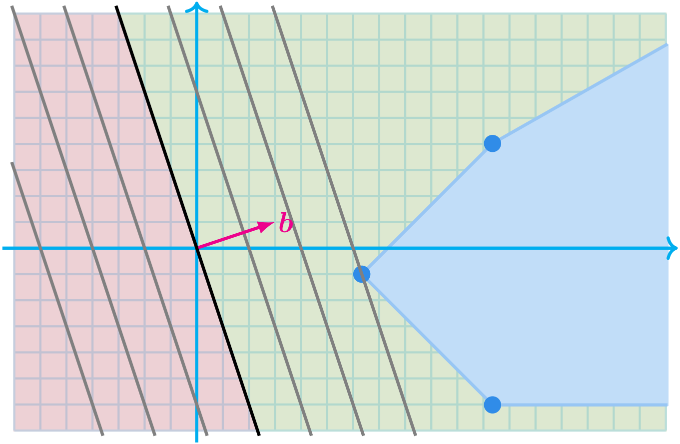


(Affine) hyperplanes



Polyhedra and faces

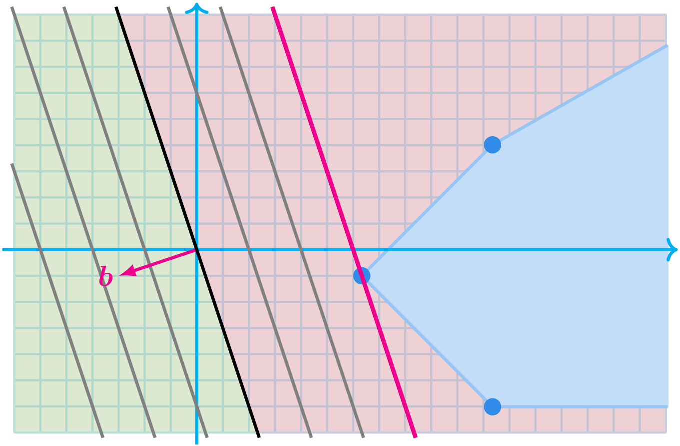
Understand where to find optima



objective:
max $\boldsymbol{b}^\mathsf{T} \cdot \boldsymbol{x}$

Maximum can be unbounded. Then, optimization is not possible

Understand where to find optima



objective:
max $\boldsymbol{b}^\intercal \cdot \boldsymbol{x}$

Maximum can be bounded. Then, all optima lie in a face

Faces of polyhedra (again)

Let P be a polyhedron.

$F \subseteq P$ is a face of P if $F = P$ or $F = P \cap H$ for some supporting hyperplane of F.

Theorem

Let $P = \{\boldsymbol{x} \in \mathbb{R}^d : A \cdot \boldsymbol{x} \geq \boldsymbol{b}\}$ and $F \subseteq \mathbb{R}^d$. The following statements are equivalent:

1. *F is a face of P.*
2. *$F = P \cap \{\boldsymbol{x} : A' \cdot \boldsymbol{x} = \boldsymbol{b}'\}$ where $A' \cdot \boldsymbol{x} \geq \boldsymbol{b}'$ is a subsystem of $A \cdot \boldsymbol{x} \geq \boldsymbol{b}$.*
3. *$F = \{\boldsymbol{x} \in P : \boldsymbol{b}^\intercal \cdot \boldsymbol{x}$ maximum achievable with points in $P\}$, for some $\boldsymbol{b} \in \mathbb{R}^d$.*

Dimension of linear (sub)spaces

Vectors $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_k$ are linearly independent whenever, for every $(a_1, \ldots, a_k) \in \mathbb{R}^k$, if $a_1 \cdot \boldsymbol{v}_1 + \cdots + a_k \cdot \boldsymbol{v}_k = \mathbf{0}$ then $(a_1, \ldots, a_k) = \mathbf{0}$.

Dimension of linear (sub)spaces

Vectors $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_k$ are linearly independent whenever, for every $(a_1, \ldots, a_k) \in \mathbb{R}^k$, if $a_1 \cdot \boldsymbol{v}_1 + \cdots + a_k \cdot \boldsymbol{v}_k = \mathbf{0}$ then $(a_1, \ldots, a_k) = \mathbf{0}$.

A linear (sub)space $S \subseteq \mathbb{R}^d$ has dimension $\dim(S) = k$ whenever

$$S = \boldsymbol{v}_1 \cdot \mathbb{R} + \cdots + \boldsymbol{v}_k \cdot \mathbb{R}$$

for some linearly independent non-zero vectors $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_k$.

Dimension of linear (sub)spaces

Vectors $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_k$ are linearly independent whenever, for every $(a_1, \ldots, a_k) \in \mathbb{R}^k$, if $a_1 \cdot \boldsymbol{v}_1 + \cdots + a_k \cdot \boldsymbol{v}_k = \mathbf{0}$ then $(a_1, \ldots, a_k) = \mathbf{0}$.

A linear (sub)space $S \subseteq \mathbb{R}^d$ has dimension $\dim(S) = k$ whenever

$$S = \boldsymbol{v}_1 \cdot \mathbb{R} + \cdots + \boldsymbol{v}_k \cdot \mathbb{R}$$

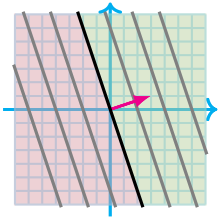
for some linearly independent non-zero vectors $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_k$.

Examples:

- $\mathbb{R}^3$ has dimension 3 : $\quad \mathbb{R}^3 = (1,0,0) \cdot \mathbb{R} + (0,1,0) \cdot \mathbb{R} + (0,0,1) \cdot \mathbb{R}$
- $\{\mathbf{0}\} \subseteq \mathbb{R}^d$ has dimension 0
- Hyperplanes in $\mathbb{R}^d$ have dimension $d-1$.

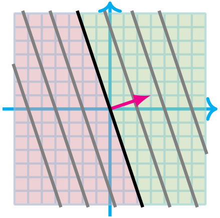
Dimension of affine (sub)spaces

A set $S \subseteq \mathbb{R}^d$ is affine whenever $S = \boldsymbol{v} + L$, where $L \subseteq \mathbb{R}^d$ is a linear subspace and $\boldsymbol{v} \in \mathbb{R}^d$.



In this case, $\dim(S) = \dim(L)$.

Dimension of affine (sub)spaces



A set $S \subseteq \mathbb{R}^d$ is affine whenever $S = \boldsymbol{v} + L$, where $L \subseteq \mathbb{R}^d$ is a linear subspace and $\boldsymbol{v} \in \mathbb{R}^d$.

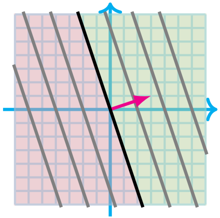
In this case, $\dim(S) = \dim(L)$.

Examples:

- $\{\mathbf{0}\} \subseteq \mathbb{R}^d$ has dimension 0
- Hyperplanes in $\mathbb{R}^d$ have dimension $d - 1$

Dimension of affine (sub)spaces

A set $S \subseteq \mathbb{R}^d$ is affine whenever $S = \boldsymbol{v} + L$, where $L \subseteq \mathbb{R}^d$ is a linear subspace and $\boldsymbol{v} \in \mathbb{R}^d$.

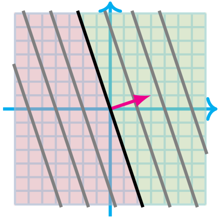


In this case, $\dim(S) = \dim(L)$.

Examples:

- points in $\mathbb{R}^d$ have dimension 0
- Hyperplanes in $\mathbb{R}^d$ have dimension $d - 1$

Dimension of affine (sub)spaces



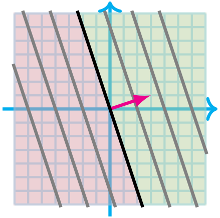
A set $S \subseteq \mathbb{R}^d$ is affine whenever $S = \boldsymbol{v} + L$, where $L \subseteq \mathbb{R}^d$ is a linear subspace and $\boldsymbol{v} \in \mathbb{R}^d$.

In this case, $\dim(S) = \dim(L)$.

Examples:

- points in $\mathbb{R}^d$ have dimension 0
- affine hyperplanes in $\mathbb{R}^d$ have dimension $d - 1$

Dimension of affine (sub)spaces



A set $S \subseteq \mathbb{R}^d$ is affine whenever $S = \boldsymbol{v} + L$, where $L \subseteq \mathbb{R}^d$ is a linear subspace and $\boldsymbol{v} \in \mathbb{R}^d$.

In this case, $\dim(S) = \dim(L)$.

Examples:

- points in $\mathbb{R}^d$ have dimension 0
- affine hyperplanes in $\mathbb{R}^d$ have dimension $d - 1$
- lines in $\mathbb{R}^d$ have dimension 1

Dimension of affine (sub)spaces

Relation between dimension and number of equations

Every affine subspace S can be characterised as $\{\boldsymbol{x} \in \mathbb{R}^d : A \cdot \boldsymbol{x} = \boldsymbol{c}\}$. Then,

$$\dim(S) + (\text{number of independent rows in } A) = d$$

Examples:

- points in $\mathbb{R}^d$ have dimension 0
- affine hyperplanes in $\mathbb{R}^d$ have dimension $d-1$
- lines in $\mathbb{R}^d$ have dimension 1

Dimension of affine (sub)spaces

Relation between dimension and number of equations

Every affine subspace S can be characterised as $\{\boldsymbol{x} \in \mathbb{R}^d : A \cdot \boldsymbol{x} = \boldsymbol{c}\}$. Then,

$$\dim(S) + (\text{number of independent rows in } A) = d$$

Examples:

- points in $\mathbb{R}^d$ have dimension 0 $\rightarrow$ d equations
- affine hyperplanes in $\mathbb{R}^d$ have dimension $d-1$ $\rightarrow$ 1 equation
- lines in $\mathbb{R}^d$ have dimension 1 $\rightarrow$ $d-1$ equations

Dimension of polyhedra

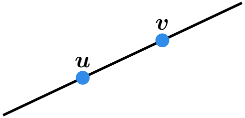
Given $S \subseteq \mathbb{R}^d$, the affine hull $\operatorname{aff}(S)$ is the smallest affine subspace containing S. Equivalently, it is the set of all affine combinations of elements in S, i.e.,

$$\operatorname{aff}(S) = \left\{ \sum_{i=1}^{k} a_i \cdot \boldsymbol{v}_i \;:\; k > 0,\ \boldsymbol{v}_i \in S,\ a_i \in \mathbb{R},\ \sum_{i=1}^{k} a_i = 1 \right\}$$

Dimension of polyhedra

Given $S \subseteq \mathbb{R}^d$, the affine hull $\operatorname{aff}(S)$ is the smallest affine subspace containing S. Equivalently, it is the set of all affine combinations of elements in S, i.e.,

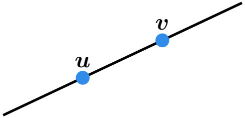
$$\operatorname{aff}(S) = \left\{ \sum_{i=1}^{k} a_i \cdot \boldsymbol{v}_i \ : \ k > 0,\ \boldsymbol{v}_i \in S,\ a_i \in \mathbb{R},\ \sum_{i=1}^{k} a_i = 1 \right\}$$

$\operatorname{aff}\{\boldsymbol{u}, \boldsymbol{v}\} =$ 

Dimension of polyhedra

Given $S \subseteq \mathbb{R}^d$, the affine hull $\operatorname{aff}(S)$ is the smallest affine subspace containing S. Equivalently, it is the set of all affine combinations of elements in S, i.e.,

$$\operatorname{aff}(S) = \left\{ \sum_{i=1}^{k} a_i \cdot \boldsymbol{v}_i \ : \ k > 0,\ \boldsymbol{v}_i \in S,\ a_i \in \mathbb{R},\ \sum_{i=1}^{k} a_i = 1 \right\}$$

$\operatorname{aff}\{\boldsymbol{u}, \boldsymbol{v}\} =$ 

Let $P \subseteq \mathbb{R}^d$ be a polyhedron. The dimension of P is the dimension of $\operatorname{aff}(P)$.

Towards algorithms: representations of numbers, vectors, and matrices

Integer $n = \pm\big(a_k \cdot 2^k + \ldots + a_1 \cdot 2^1 + a_0\big) \in \mathbb{Z}$ with all $a_i \in \{0,1\}$:

size (bit length) $\quad \langle n \rangle = 2 + \lceil \log_2(|n| + 1) \rceil$

Towards algorithms: representations of numbers, vectors, and matrices

Integer $n = \pm\big(a_k \cdot 2^k + \ldots + a_1 \cdot 2^1 + a_0\big) \in \mathbb{Z}$ with all $a_i \in \{0,1\}$:

$$\text{size (bit length)} \qquad \langle n \rangle = 2 + \lceil \log_2(|n| + 1) \rceil$$

Rational $r = \frac{n}{d} \in \mathbb{Q}$ with $n \in \mathbb{Z}$, $d \in \mathbb{Z}_{>0}$, $\gcd(n,d) = 1$:

$$\langle r \rangle = \langle n \rangle + \langle d \rangle$$

Vector $\boldsymbol{v} \in \mathbb{Q}^m$:

$$\langle \boldsymbol{v} \rangle = \sum_{i=1}^{m} \langle v_i \rangle$$

Matrix $M = (m_{ij}) \in \mathbb{Q}^{k \times \ell}$:

$$\langle M \rangle = \sum_{i=1}^{k} \sum_{j=1}^{\ell} \langle m_{ij} \rangle$$

Towards algorithms: the Minkowski–Weyl theorem, revisited

Theorem (Minkowski–Weyl; 1897, 1935)

Consider $S \subseteq \mathbb{R}^d$. The two following statements are equivalent:

(H) $S = \{\boldsymbol{x} \in \mathbb{R}^d : A \cdot \boldsymbol{x} \geq \boldsymbol{b}\}$ *for some matrix* $A \in \mathbb{Q}^{n \times d}$ *and vector* $\boldsymbol{b} \in \mathbb{Q}^d$

(V) $S = \mathrm{conv}(V) + \mathrm{cone}(W)$ *for some finite sets* $V, W \subseteq \mathbb{Q}^d$ *.*

Towards algorithms: the Minkowski–Weyl theorem, revisited

Theorem (Minkowski–Weyl; 1897, 1935)

Consider $S \subseteq \mathbb{R}^d$. The two following statements are equivalent:

(H) $S = \{\boldsymbol{x} \in \mathbb{R}^d : A \cdot \boldsymbol{x} \geq \boldsymbol{b}\}$ *for some matrix $A \in \mathbb{Q}^{n \times d}$ and vector $\boldsymbol{b} \in \mathbb{Q}^d$*

(V) $S = \mathrm{conv}(V) + \mathrm{cone}(W)$ *for some finite sets $V, W \subseteq \mathbb{Q}^d$.*

Cost of switching, for both directions:

Bit length of numbers:

$$\langle \texttt{output} \rangle \leq \mathsf{poly}(d) \cdot \langle \texttt{input} \rangle$$

Amount of numbers (with repetitions):

$$\#(\texttt{output}) \leq \#(\texttt{input})^{\mathrm{poly}(d)}$$

Towards algorithms: the Minkowski–Weyl theorem, revisited

Theorem (Minkowski–Weyl; 1897, 1935)

Consider $S \subseteq \mathbb{R}^d$. The two following statements are equivalent:

(H) $S = \{\boldsymbol{x} \in \mathbb{R}^d : A \cdot \boldsymbol{x} \geq \boldsymbol{b}\}$ *for some matrix* $A \in \mathbb{Q}^{n \times d}$ *and vector* $\boldsymbol{b} \in \mathbb{Q}^d$

(V) $S = \mathrm{conv}(V) + \mathrm{cone}(W)$ *for some finite sets* $V, W \subseteq \mathbb{Q}^d$ *.*

Cost of switching, (V) $\to$ (H):

Bit length of numbers:

$$\langle A \rangle, \langle \boldsymbol{b} \rangle \leq \mathsf{poly}(d) \cdot \max(\langle V \rangle, \langle W \rangle)$$

Amount of numbers (with repetitions):

$$\#[A \,|\, \boldsymbol{b}] \leq (\#V + \#W)^{\mathrm{poly}(d)}$$

Towards algorithms: the Minkowski–Weyl theorem, revisited

Theorem (Minkowski–Weyl; 1897, 1935)

Consider $S \subseteq \mathbb{R}^d$. The two following statements are equivalent:

(H) $S = \{\boldsymbol{x} \in \mathbb{R}^d : A \cdot \boldsymbol{x} \geq \boldsymbol{b}\}$ *for some matrix $A \in \mathbb{Q}^{n \times d}$ and vector $\boldsymbol{b} \in \mathbb{Q}^d$*

(V) $S = \mathrm{conv}(V) + \mathrm{cone}(W)$ *for some finite sets $V, W \subseteq \mathbb{Q}^d$.*

Cost of switching, for both directions:

Bit length of numbers:

$$\langle \texttt{output} \rangle \leq \mathsf{poly}(d) \cdot \langle \texttt{input} \rangle$$

$\leftarrow$ deciding the non-emptiness of S is a $\mathbf{NP}$ problem!

Amount of numbers (with repetitions):

$$\#(\texttt{output}) \leq \#(\texttt{input})^{\mathrm{poly}(d)}$$

$\mathbf{PTIME}$: problems with efficient algorithms

$$\mathrm{DTIME}(n^k) = \text{ solution computable in } O(n^k) \text{ on a deterministic Turing machine}$$

$$\mathbf{PTIME} = \bigcup_{k \geq 1} \mathrm{DTIME}(n^k)$$

Cobham–Edmonds Thesis (1965)

Efficiently computable in a reasonable computational model

$=$

Computable in polynomial time on a deterministic Turing machine

$\mathbf{NP}$: problems with efficiently verifiable solutions

Decision problem: problem with a yes/no answer.

The complexity class $\mathbf{NP}$ contains all decision problems where, for each input x:

- if the answer is "yes", there exists a **certificate** y of small size: $\langle y \rangle \leq \mathrm{poly}(\langle x \rangle)$;

  whether the certificate is valid can be checked in time $\mathrm{poly}(\langle x \rangle, \langle y \rangle)$; and
- if the answer is "no", there is no such witness.

Simplex algorithm

Standard form of linear programs

$$\begin{aligned} &\max_{\boldsymbol{x} \in \mathbb{R}^d} \ \boldsymbol{c}^\mathsf{T} \cdot \boldsymbol{x} \\ &\text{subject to } A \cdot \boldsymbol{x} \geq \boldsymbol{b} \end{aligned} \quad \longrightarrow \quad \begin{aligned} &\max_{\boldsymbol{y} \in \mathbb{R}^d} \ \boldsymbol{c}^\mathsf{T} \cdot \boldsymbol{y} \\ &\text{subject to } A' \cdot \boldsymbol{y} = \boldsymbol{b}' \\ &\qquad\qquad\quad \boldsymbol{y} \geq \boldsymbol{0} \end{aligned}$$

Standard form of linear programs

$$\begin{aligned} \underset{\boldsymbol{x}\in\mathbb{R}^d}{\text{maximize}} \quad & \boldsymbol{c}^\mathsf{T} \cdot \boldsymbol{x} \\ \text{subject to} \quad & A \cdot \boldsymbol{x} \geq \boldsymbol{b} \end{aligned} \quad \longrightarrow \quad \begin{aligned} \underset{\boldsymbol{y}\in\mathbb{R}^d}{\text{maximize}} \quad & \boldsymbol{c}^\mathsf{T} \cdot \boldsymbol{y} \\ \text{subject to} \quad & A' \cdot \boldsymbol{y} = \boldsymbol{b}' \\ & \boldsymbol{y} \geq \boldsymbol{0} \end{aligned}$$

Transformation:

1. Substitute the every row $\boldsymbol{a}^\mathsf{T} \cdot \boldsymbol{x} \geq \boldsymbol{b}$ with $\boldsymbol{a}^\mathsf{T} \cdot \boldsymbol{x} - s = \boldsymbol{b}$ (where s is a new variable).
2. Substitute each x with $y^+ - y^-$ (where y^+ and y^- are new variables).
3. For every new variable z introduced above, add the constraint $z \geq 0$.
4. Rewrite every row $\boldsymbol{a}^\mathsf{T} \cdot \boldsymbol{x} = b$ with b negative to $-\boldsymbol{a}^\mathsf{T} \cdot \boldsymbol{x} = -b$.

Solving linear programs

Simplex algorithm on a high level:

- Start at some initial vertex of **polytope**
- Move to neighbor vertex improving objective function if it exists
- Otherwise return current vertex

Solving linear programs

Simplex algorithm on a high level:

- Start at some initial vertex of **polytope**
- Move to neighbor vertex improving objective function if it exists
- Otherwise return current vertex

Historical remarks:

- Developed by George Dantzig in 1947
- One of the ten most important algorithms of the 20th century

Simplex in detail

Vertices:

- Point $\boldsymbol{v} \in \mathbb{R}^d$ is vertex of polytope only if it satisfies d defining inequality constraints with equality
- Point $\boldsymbol{w}$ is neighbor of $\boldsymbol{v}$ if $\boldsymbol{v}$ and $\boldsymbol{w}$ share $d-1$ defining inequalities

Simplex is simple when starting at origin:

- If objective function is non-positive we are done
- Otherwise increase some variable with positive coefficient until a constraint becomes tight (pivot rule)

Example

$$
\begin{aligned}
\text{maximize } & x_1 + 6x_2 \\
\text{s.t. } & x_1 \leq 2 \\
& x_2 \leq 3 \\
& x_1 + x_2 \leq 4 \\
& x_1 \geq 0 \\
& x_2 \geq 0
\end{aligned}
$$

Example

$$
\begin{aligned}
\text{maximize}\ & x_1 + 6x_2 \\
\text{s.t.}\ & x_1 \leq 2 \\
& x_2 \leq 3 \\
& x_1 + x_2 \leq 4 \\
& x_1 \geq 0 \\
& x_2 \geq 0
\end{aligned}
$$

Example

$$
\begin{aligned}
\text{maximize } & x_1 + 6x_2 \\
\text{s.t. } & x_1 \leq 2 \\
& x_2 \leq 3 \\
& x_1 + x_2 \leq 4 \\
& x_1 \geq 0 \\
& x_2 \geq 0
\end{aligned}
$$

Increasing x_2 from 0 to 3:

- Increases objective function to 18
- Leads to new neighbor vertex $\boldsymbol{w} = (0, 3)$

Example

$$\begin{aligned} \text{maximize } & x_1 + 6x_2 \\ \text{s.t. } & x_1 \leq 2 \\ & {\color{red} x_2 \leq 3} \\ & x_1 + x_2 \leq 4 \\ & {\color{red} x_1 \geq 0} \\ & x_2 \geq 0 \end{aligned}$$

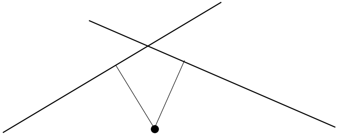
Increasing x_2 from 0 to 3:

- Increases objective function to 18
- Leads to new neighbor vertex $\boldsymbol{w} = (0, 3)$

Translating new vertex to origin

Turn new vertex into new origin:

- Any point inside polytope uniquely definable in terms of distances from defining hyperplanes



- Slack (distance) of point in polytope to i-th hyperplane $\boldsymbol{a}_i \cdot \boldsymbol{x} \leq b_i$ given by $y_i = b_i - \boldsymbol{a}_i \cdot \boldsymbol{x}$ and $y_i \geq 0$
- Yields d equations in d unknowns
- Express every x_i in terms of $y_1, \ldots, y_d$ and substitute

Example

$$
\begin{aligned}
\text{maximize}\ & x_1 + 6x_2 \\
\text{s.t.}\ & x_1 \leq 2 \\
& x_2 \leq 3 \\
& x_1 + x_2 \leq 4 \\
& x_1 \geq 0 \\
& x_2 \geq 0
\end{aligned}
$$

Example

$$
\begin{aligned}
\text{maximize } & x_1 + 6x_2 \\
\text{s.t. } & x_1 \le 2 \\
& \color{red}{x_2 \le 3} \\
& x_1 + x_2 \le 4 \\
& \color{red}{x_1 \ge 0} \\
& x_2 \ge 0
\end{aligned}
$$

$$
\begin{aligned}
y_1 &= x_1 & \rightsquigarrow && x_1 &= y_1 \\
y_2 &= 3 - x_2 & \rightsquigarrow && x_2 &= 3 - y_2
\end{aligned}
$$

Example

$$\begin{aligned} \text{maximize } & x_1 + 6x_2 \\ \text{s.t. } & x_1 \leq 2 \\ & x_2 \leq 3 \\ & x_1 + x_2 \leq 4 \\ & x_1 \geq 0 \\ & x_2 \geq 0 \end{aligned}$$

$$\begin{aligned} \text{maximize } & 18 + y_1 - 6y_2 \\ \text{s.t. } & y_1 \leq 2 \\ & y_2 \geq 0 \\ & y_1 - y_2 \leq 1 \\ & y_1 \geq 0 \\ & y_2 \leq 3 \end{aligned}$$

$$y_1 = x_1 \quad \rightsquigarrow \quad x_1 = y_1$$

$$y_2 = 3 - x_2 \quad \rightsquigarrow \quad x_2 = 3 - y_2$$

Example

$$\begin{aligned} \text{maximize } & x_1 + 6x_2 \\ \text{s.t. } & x_1 \leq 2 \\ & {\color{red} x_2 \leq 3} \\ & x_1 + x_2 \leq 4 \\ & {\color{red} x_1 \geq 0} \\ & x_2 \geq 0 \end{aligned}$$

$$\begin{aligned} \text{maximize } & 18 + y_1 - 6y_2 \\ \text{s.t. } & y_1 \leq 2 \\ & y_2 \geq 0 \\ & y_1 - y_2 \leq 1 \\ & y_1 \geq 0 \\ & y_2 \leq 3 \\ & y_1 \geq 0 \\ & y_2 \geq 0 \end{aligned}$$

$$\begin{aligned} y_1 &= x_1 & \rightsquigarrow \quad x_1 &= y_1 \\ y_2 &= 3 - x_2 & \rightsquigarrow \quad x_2 &= 3 - y_2 \end{aligned}$$

Finding a vertex to start from

Simplex can be used to find initial vertex. Given

$$\begin{aligned} \underset{\boldsymbol{x}\in\mathbb{R}^d}{\text{maximize}}\, & \boldsymbol{c}\cdot\boldsymbol{x} \\ \text{s.t. } & A\cdot\boldsymbol{x} = \boldsymbol{b} \\ & \boldsymbol{x} \geq \boldsymbol{0} \end{aligned}$$

Finding a vertex to start from

Simplex can be used to find initial vertex. Given

$$\begin{aligned}\underset{\boldsymbol{x}\in\mathbb{R}^d}{\text{maximize}}\, \boldsymbol{c}\cdot\boldsymbol{x} \\ \text{s.t. } A\cdot\boldsymbol{x} = \boldsymbol{b} \\ \boldsymbol{x} \geq \boldsymbol{0}\end{aligned}$$

To find initial vertex:

- Augment i-th row of system with fresh variable z_i, $z_i \geq 0$
- Change objective function to $-(z_1 + \cdots + z_m)$
- New system has initial vertex $z_i := b_i$, $x_j := 0$
- Run simplex, two possible outcomes
  - Solution with objective 0 $\rightsquigarrow$ gives vertex of original system
  - Otherwise original system infeasible

Ellipsoid method

Ellipsoid method for LP (Khachiyan, 1979)

Ellipsoid method for LP (Khachiyan, 1979)

1. Reduce optimality to feasibility

2. Find a feasible solution if one exists

Ellipsoid method for LP (Khachiyan, 1979)

1. Reduce optimality to feasibility

2. Find a feasible solution if one exists

Insight: Polyhedra live in a discrete world!

Ellipsoid method for LP (Khachiyan, 1979)

1. Reduce optimality to feasibility
2. Find a feasible solution if one exists

Insight: Polyhedra live in a discrete world!

How to find a feasible solution?

1. Choose a $\rho > 0$ s.t. all vertices of P are in $E = \{\boldsymbol{x}^2 \leq \rho^2\}$.

2. `while(true):`

   Let z be the center of E. If z is feasible, stop.
   Otherwise find a violated constraint (use separation oracle).

   $E' \leftarrow$ smallest ellipsoid containing $E \cap \{\boldsymbol{x} : \boldsymbol{a}_i \cdot \boldsymbol{x} \geq c_i\}$,
   where $\boldsymbol{a}_i \cdot \boldsymbol{x} \geq c_i$ is the violated constraint.
   $E \leftarrow E'$.

   If $\mathrm{vol}(E') \leq$ magic number, stop with "infeasible".

Is this efficient?

Theorem

The number of iterations of the ellipsoid method is polynomial in n and s, the maximum size of numbers in the system $A \cdot \boldsymbol{x} \geq \boldsymbol{c}$.

Is this efficient?

Theorem

The number of iterations of the ellipsoid method is polynomial in n and s, the maximum size of numbers in the system $A \cdot \boldsymbol{x} \geq \boldsymbol{c}$.

Lemma 1: $\mathrm{vol}(E') \leq \mathrm{vol}(E) \cdot \left(1 - \frac{1}{\mathrm{poly}(n,s)}\right)$.

Lemma 2: If P is full-dimensional, then $\mathrm{vol}(P) \geq 2^{-\mathrm{poly}(n,s)}$.

Lemma 3: ρ can be chosen as $2^{\mathrm{poly}(n,s)}$.

Lemma 4: "Magic number" can be chosen as $2^{-\mathrm{poly}(n,s)}$.

Caveats

1. We assumed that $\mathrm{vol}(P) > 0$ if $P \neq \varnothing$.
2. We assumed unit-cost arithmetic.

Caveats

1. We assumed that $\mathrm{vol}(P) > 0$ if $P \neq \varnothing$.
2. We assumed unit-cost arithmetic.

Neither assumption is necessary.

Conclusion
Linear programming is in $\mathbf{PTIME}$.

Integer programming

Integer programming

Optimization:

Input: matrix $A \in \mathbb{Z}^{m \times d}$, vectors $\boldsymbol{c} \in \mathbb{Z}^m$ and $\boldsymbol{b} \in \mathbb{Z}^d$
Output: a vector $\boldsymbol{x} \in \mathbb{Z}^d$ that maximizes $\boldsymbol{b} \cdot \boldsymbol{x}$ and satisfies $A \cdot \boldsymbol{x} \geq \boldsymbol{c}$

Feasibility:

Input: matrix $A \in \mathbb{Z}^{m \times d}$, vector $\boldsymbol{c} \in \mathbb{Z}^m$
Output: does there exist an $\boldsymbol{x} \in \mathbb{Z}^d$ that satisfies $A \cdot \boldsymbol{x} \geq \boldsymbol{c}$?

Integer programming

$$\begin{aligned} \text{maximize } & \boldsymbol{b}^{\intercal} \cdot \boldsymbol{x} \\ \text{s.t } & A \cdot \boldsymbol{x} \geq \boldsymbol{c}, \\ & \boldsymbol{x} \in \mathbb{Z}^d \end{aligned}$$

Very powerful formalism for encoding combinatorial questions.

Geometry of integer programming

$\mathbb{N}^d$: Linear, hybrid linear, and semi-linear sets

[Parikh (1961)]

$\mathbb{N}^d$: Linear, hybrid linear, and semi-linear sets

[Parikh (1961)]

$$\left.\begin{array}{l}\text{Vectors } \boldsymbol{b} \text{ in } B\text{: base vectors} \\ \text{Vectors } \boldsymbol{p}_i \text{ in } P\text{: period vectors}\end{array}\right\} \text{generators}$$

Linear set:

$$L(\boldsymbol{b}, P) = \{\boldsymbol{b} + \lambda_1 \boldsymbol{p}_1 + \ldots \lambda_s \boldsymbol{p}_s : \\ \boldsymbol{p}_1, \ldots, \boldsymbol{p}_s \in P,\ \lambda_1, \ldots, \lambda_s \in \mathbb{N},\ s \geq 0\}$$

Hybrid linear set:

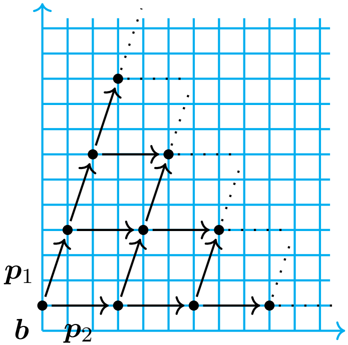
$$L(B, P) = \bigcup_{\boldsymbol{b} \in B} L(\boldsymbol{b}, P)$$

Semi-linear set:

$$M = \bigcup_{i \in I} L(B_i, P_i)$$

$\mathbb{N}^d$: Linear, hybrid linear, and semi-linear sets

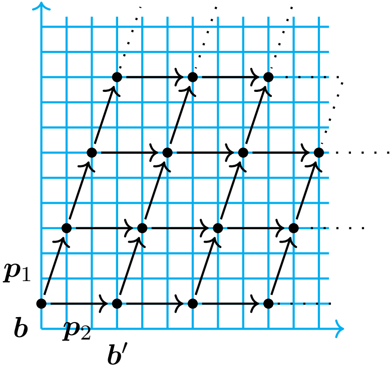
[Parikh (1961)]



Linear $<$ Hybrid linear $<$ Semi-linear

$\mathbb{N}^d$: Linear, hybrid linear, and semi-linear sets

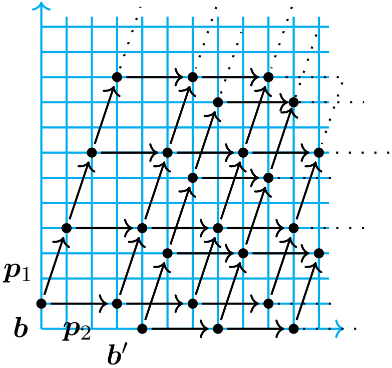
[Parikh (1961)]



Linear $<$ Hybrid linear $<$ Semi-linear

$\mathbb{N}^d$: Linear, hybrid linear, and semi-linear sets

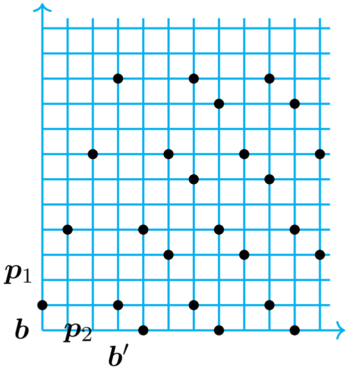
[Parikh (1961)]



Linear $<$ Hybrid linear $<$ Semi-linear

$\mathbb{N}^d$: Linear, hybrid linear, and semi-linear sets

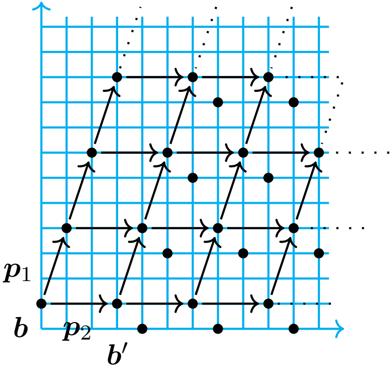
[Parikh (1961)]



Linear $<$ Hybrid linear $<$ Semi-linear

$\mathbb{N}^d$: Linear, hybrid linear, and semi-linear sets

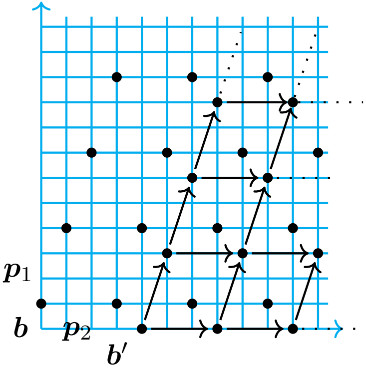
[Parikh (1961)]



Linear $<$ Hybrid linear $<$ Semi-linear

$\mathbb{N}^d$: Linear, hybrid linear, and semi-linear sets

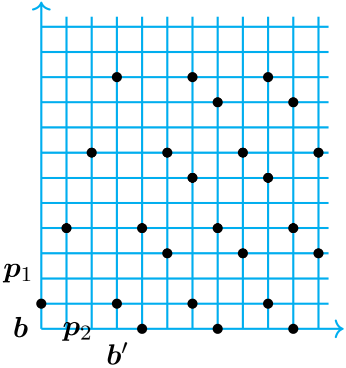
[Parikh (1961)]



Linear $<$ Hybrid linear $<$ Semi-linear

$\mathbb{N}^d$: Linear, hybrid linear, and semi-linear sets

[Parikh (1961)]



Linear $<$ Hybrid linear $<$ Semi-linear

$$\left\{ \sum \lambda_i \boldsymbol{b}_i + \sum \mu_j \boldsymbol{p}_j : \quad \sum \lambda_i = 1,\ \lambda_i \geq 0,\ \mu_j \geq 0 \right\}$$

$$\left\{\sum \lambda_i \boldsymbol{b}_i + \sum \mu_j \boldsymbol{p}_j : \quad \sum \lambda_i = 1,\ \lambda_i \geq 0,\ \mu_j \geq 0\right\}$$

- $\lambda_i, \mu_j \in \mathbb{R}$: (rational) convex polyhedron $\mathrm{conv} B + \mathrm{cone} P$
- $\lambda_i, \mu_j \in \mathbb{Z}$: hybrid linear set $L(B, P)$

Hybrid linear sets are "discrete convex polyhedra"!

$$\left\{\sum \lambda_i \boldsymbol{b}_i + \sum \mu_j \boldsymbol{p}_j : \quad \sum \lambda_i = 1,\ \lambda_i \geq 0,\ \mu_j \geq 0\right\}$$

- $\lambda_i, \mu_j \in \mathbb{R}$: (rational) convex polyhedron $\mathrm{conv} B + \mathrm{cone} P$
- $\lambda_i, \mu_j \in \mathbb{Z}$: hybrid linear set $L(B, P)$

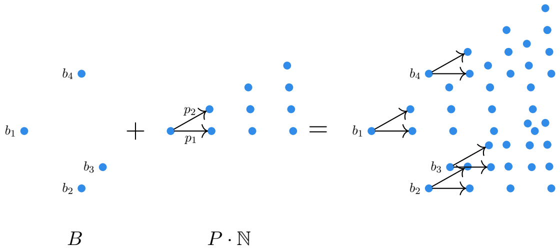
The geometry of a system of inequalities over the **integers**

Theorem (von zur Gathen & Sieveking, '78)

Consider $S \subseteq \mathbb{Z}^d$. Then, below (H) implies (V), but not vice versa:

(H) $S = \{\boldsymbol{x} \in \mathbb{Z}^d : A \cdot \boldsymbol{x} \leq \boldsymbol{c}\}$ *for some* $A \in \mathbb{Z}^{n \times d}$ *and* $\boldsymbol{c} \in \mathbb{Z}^m$

(V) $S = L(B, P)$ *for some finite sets* $B, P \subseteq \mathbb{Z}^d$.



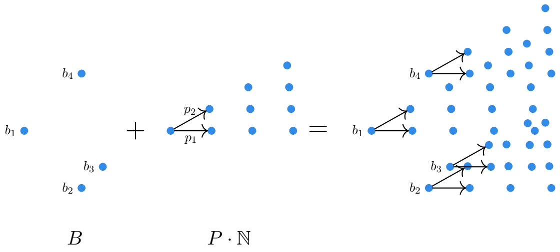
The geometry of a system of inequalities over the **integers**

Theorem (von zur Gathen & Sieveking, '78)

Consider $S \subseteq \mathbb{Z}^d$. Then, below (H) implies (V), but not vice versa:

(H) $S = \{\boldsymbol{x} \in \mathbb{Z}^d : A \cdot \boldsymbol{x} \leq \boldsymbol{c}\}$ *for some* $A \in \mathbb{Z}^{n \times d}$ *and* $\boldsymbol{c} \in \mathbb{Z}^m$

(V) $S = L(B, P)$ *for some finite sets* $B, P \subseteq \mathbb{Z}^d$.



- The blowup in size can be exponential.
- The size of all numbers stays polynomial.

Corollary: Small solutions

Corollary

If a system of inequalities $A \cdot \boldsymbol{x} \geq \boldsymbol{c}$ has a solution in $\mathbb{Z}^d$,
then it has one where numbers have **size** *at most* $\mathrm{poly}(\langle A \rangle, \langle \boldsymbol{c} \rangle)$
(that is, $x_i \leq 2^{\mathrm{poly}(\langle A \rangle, \langle \boldsymbol{c} \rangle)}$).

Computational complexity of integer programming

Let IP denote the decision problem for integer programming.

Theorem

IP *is* $\mathbf{NP}$*-complete.*

Computational complexity of integer programming

Let IP denote the decision problem for integer programming.

Theorem

IP *is* $\mathbf{NP}$*-complete.*

Proof:

- $\mathbf{NP}$-hardness: by reduction from SAT.
- Membership in $\mathbf{NP}$: from existence of small solutions.

Branch and bound

Branch and bound for integer programming

[Dakin (1965), Land and Doig (1960)]

Assume that $P = \{\boldsymbol{x} \in \mathbb{R}^d \colon A \cdot \boldsymbol{x} \geq \boldsymbol{c}\}$ is bounded.

Start with $\Pi = \big\{P\big\}$. `while(true):`

1. Given $\Pi = \big\{P_1, \ldots, P_k\big\}$, determine
$$\mu_j = \max_{\boldsymbol{x} \in P_j} \boldsymbol{b} \cdot \boldsymbol{x}.$$
(If all $\mu_j = -\infty$, return "not feasible".) Pick P_j that has the largest μ_j; let $\boldsymbol{x}^*$ be the optimal solution in P_j ($\boldsymbol{b} \cdot \boldsymbol{x}^* = \mu_j$).
2. If $\boldsymbol{x}^*$ is integral, it is optimal for P.
3. Otherwise, let x_i be a non-integral component, say $x_i = \zeta$.
Replace P_j in Π with $P_j' = P_j \cap \{\boldsymbol{x} \in \mathbb{R}^d \colon x_i \leq \lfloor \zeta \rfloor\}$ and
$$P_j'' = P_j \cap \{\boldsymbol{x} \in \mathbb{R}^d \colon x_i \geq \lceil \zeta \rceil\}.$$

Branch and bound for integer programming

Proposition

If P is bounded, the branch and bound method terminates in at most exponentially many steps with a correct answer.

Branch and bound for integer programming

Proposition

If P is bounded, the branch and bound method terminates in at most exponentially many steps with a correct answer.

If P is unbounded:

1. Apply the method to
$$P' := P \cap \left\{ \boldsymbol{x} \in \mathbb{R}^d \colon x_i \leq 2^{\mathrm{poly}(\langle A \rangle, \langle \boldsymbol{c} \rangle)} \text{ for all } i \right\}.$$
2. If there is no solution in P', there is none in P either.
3. Otherwise check if $\max\{\boldsymbol{b} \cdot \boldsymbol{x} \colon \boldsymbol{x} \in P\} = +\infty$.
   - If yes, then the maximum over integers is also $+\infty$.
   - If no, then the maximum over integers is attained inside P'.

Randomized rounding

Randomized rounding is a technique for solving combinatorial problems that have nice encodings as integer programs.

Set cover problem

Input: finite sets U and $S_1, \ldots, S_m \subseteq U$
Output: set $I \subseteq \{1, \ldots, m\}$ of minimum cardinality
such that $\bigcup_{i \in I} S_i = U$

Claim

The set cover problem is $\mathbf{NP}$-complete.

Integer program for set cover

$$
\begin{aligned}
\text{minimize} \quad & \sum_{i=1}^{m} x_i \\
\textsf{subject to} \quad & \sum_{j:\, i \in S_j} x_j \geq 1, && i = 1, \ldots, n, \\
& x_i \geq 0, && i = 1, \ldots, m, \\
& x_i \in \mathbb{Z}, && i = 1, \ldots, m
\end{aligned}
$$

LP relaxation of the integer program

$$
\begin{aligned}
\text{minimize} \quad & \sum_{i=1}^{m} x_i \\
\textsf{subject to} \quad & \sum_{j\,:\, i \in S_j} x_j \geq 1, && i = 1, \ldots, n, \\
& x_i \geq 0, && i = 1, \ldots, m, \\
& x_i \in \mathbb{R}, && i = 1, \ldots, m
\end{aligned}
$$

Randomized rounding for set cover

Suppose $|U| = n$. Let $\boldsymbol{x}^*$ be the optimal fractional cover.

- $C \leftarrow \varnothing$.
- For all $i = 1, \ldots, n$:
  repeat $c \ln n$ times: flip a coin with success probability x_i^*;
  if at least one success, add i to C.
- Return C if it is a cover.

Randomized rounding for set cover

Suppose $|U| = n$. Let $\boldsymbol{x}^*$ be the optimal fractional cover.

- $C \leftarrow \varnothing$.

- For all $i = 1, \ldots, n$:
  repeat $c \ln n$ times: flip a coin with success probability x_i^*;
  if at least one success, add i to C.

- Return C if it is a cover.

Theorem

If $c \geq 2$, this algorithm produces a cover with probability at least $1 - 1/\mathrm{poly}(n)$. The expected size of C, conditioned on C being a cover, is at most $O(\log n)$ times the size of a smallest cover.

Summary of today's lecture

We have seen two algorithms for Linear programming:

- Simplex algorithm
- Ellipsoid method

We saw two algorithms for Integer Linear programming:

- Branch-and-bound
- Randomized rounding

We also saw a Minkowski-Weyl analogue for integer polytopes:

- Linear sets, hybrid linear sets, and semi-linear sets

Agenda

Starting from tomorrow, we move to first-order arithmetic theories!

Tomorrow	Quantifier elimination procedures
Thursday	Automata-based procedures
Friday	Geometric procedures