Linear arithmetic: Geometry, algorithms, and logic

Wednesday

Dmitry Chistikov Christoph Haase

Centre for Discrete Mathematics and its Applications (DIMAP) &
Department of Computer Science, University of Warwick, UK

Department of Computer Science
University of Oxford, UK

ESSLLI 2018

Today's lecture

- Linear programming: Ellipsoid method
- Integer programming
  - Geometry: Hybrid linear sets
  - Computational complexity: $\mathbf{NP}$-completeness
  - Algorithms: Branch and bound, randomized rounding

Ellipsoid method

Computational complexity: Decision problems and algorithms

Decision problem:
$X \subseteq \{0,1\}^*$ (encodings of yes-instances)

Algorithm for X:
says "yes" on every $x \in X$, "no" on every $x \in \{0,1\}^* \setminus X$

Time complexity

- of algorithm $\mathcal{A}$ on input x
- of algorithm $\mathcal{A}$ on inputs of length n (worst-case)
- of decision problem X

Complexity class $\mathbf{P}$ and efficient algorithms

Cobham–Edmonds thesis (1965):

Efficiently computable in a reasonable computational model

=

Computable in polynomial time on a Turing machine

$$\mathbf{P} = \bigcup_{d \geq 1} \bigcup_{c \geq 1} \mathrm{DTIME}(c \cdot n^d)$$

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 $E \cap \{\boldsymbol{x} : \boldsymbol{a}_i \cdot \boldsymbol{x} \geq \boldsymbol{a} \cdot \boldsymbol{z}\}$,
   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 d 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 d and s, the maximum size of numbers in the system $A \cdot \boldsymbol{x} \geq \boldsymbol{c}$.

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

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

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

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

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 $\mathbf{P}$.

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

$$\begin{aligned} & \boldsymbol{b} \cdot \boldsymbol{x} \to \max \\ \text{subject to } & A \cdot \boldsymbol{x} \geq \boldsymbol{c}, \\ & \boldsymbol{x} \in \mathbb{Z}^d \end{aligned}$$

Very powerful formalism for encoding combinatorial questions.

Today's lecture

- Linear programming: Ellipsoid method
- Integer programming
  - Geometry: Hybrid linear sets
  - Computational complexity: $\mathbf{NP}$-completeness
  - Algorithms: Branch and bound, randomized rounding

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)]

Vectors $\boldsymbol{b}$ in B: base vectors
Vectors $\boldsymbol{p}_i$ in P: period vectors
} 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:

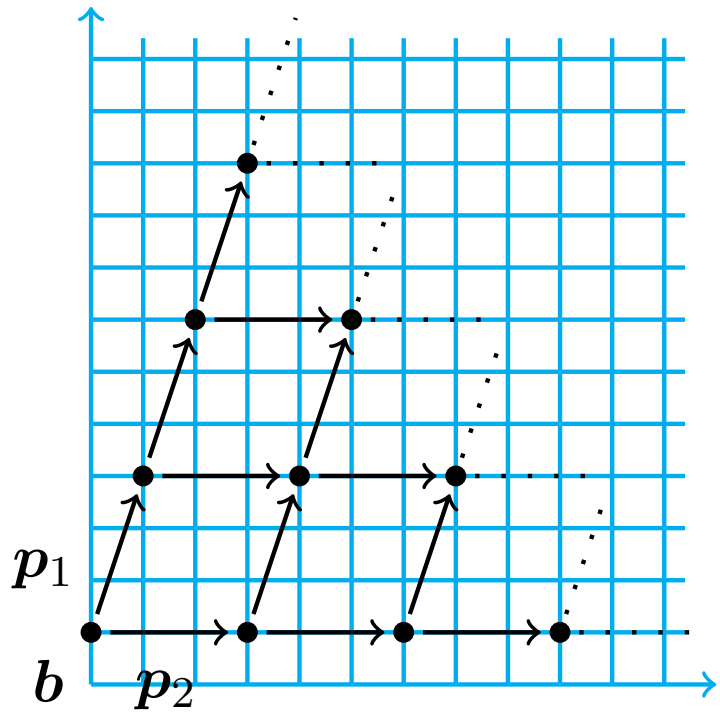
$$L(B, P) = \bigcup_{\boldsymbol{b} \in B} L(\boldsymbol{b}, P)$$

Semi-linear set:

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

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

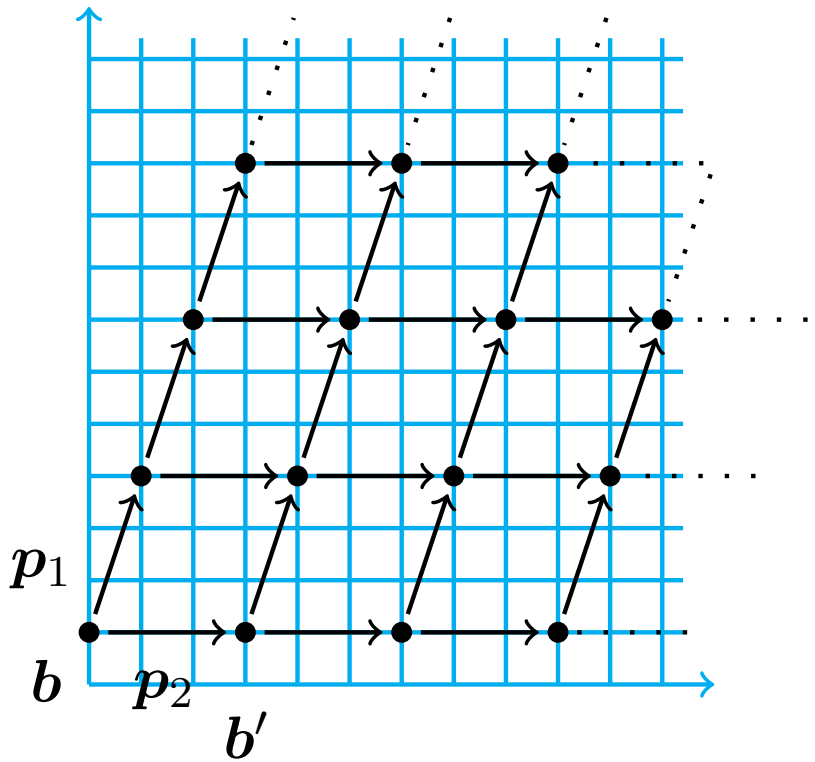
[Parikh (1961)]



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

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

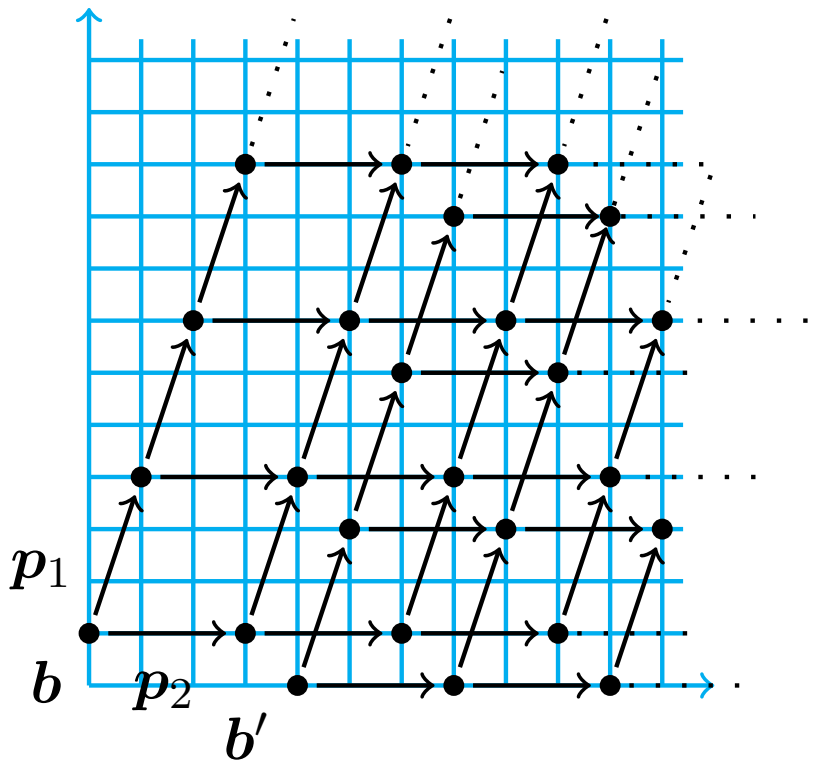
[Parikh (1961)]



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

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

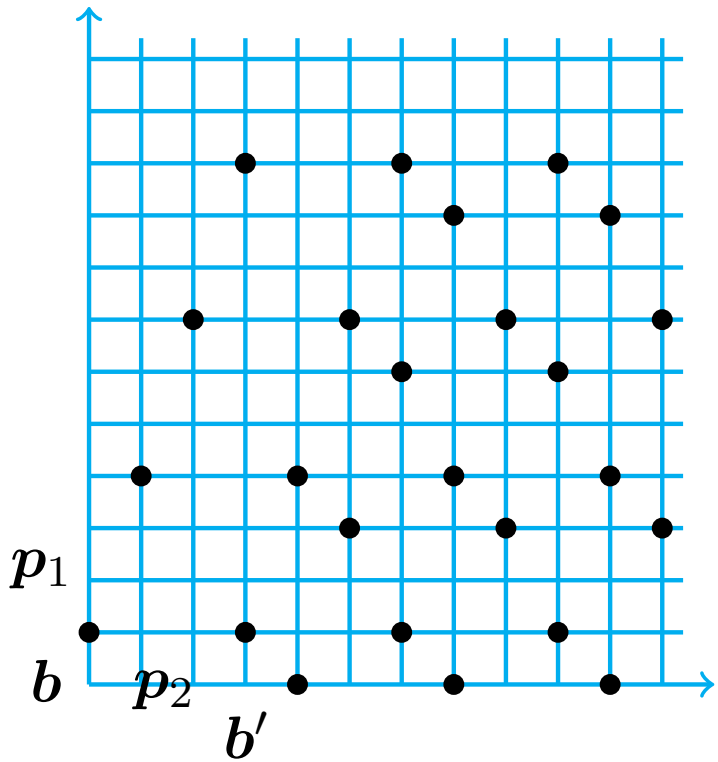
[Parikh (1961)]



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

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

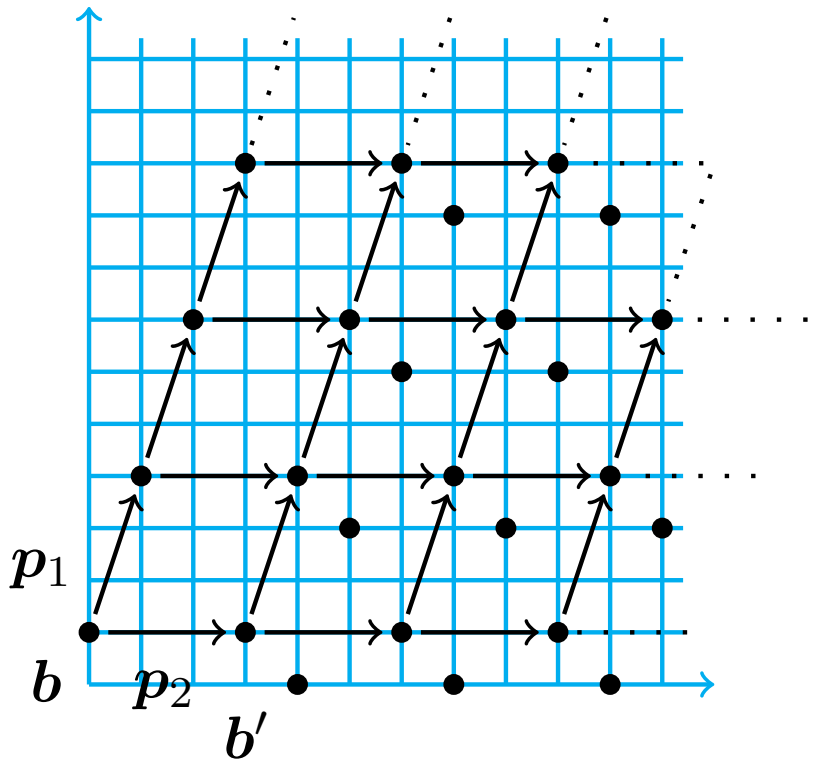
[Parikh (1961)]



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

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

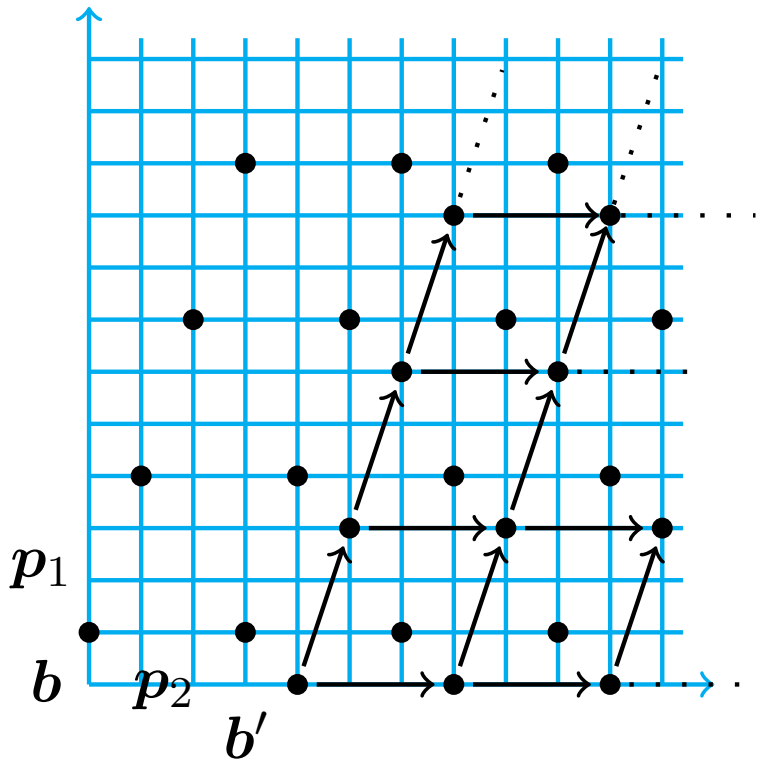
[Parikh (1961)]



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

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

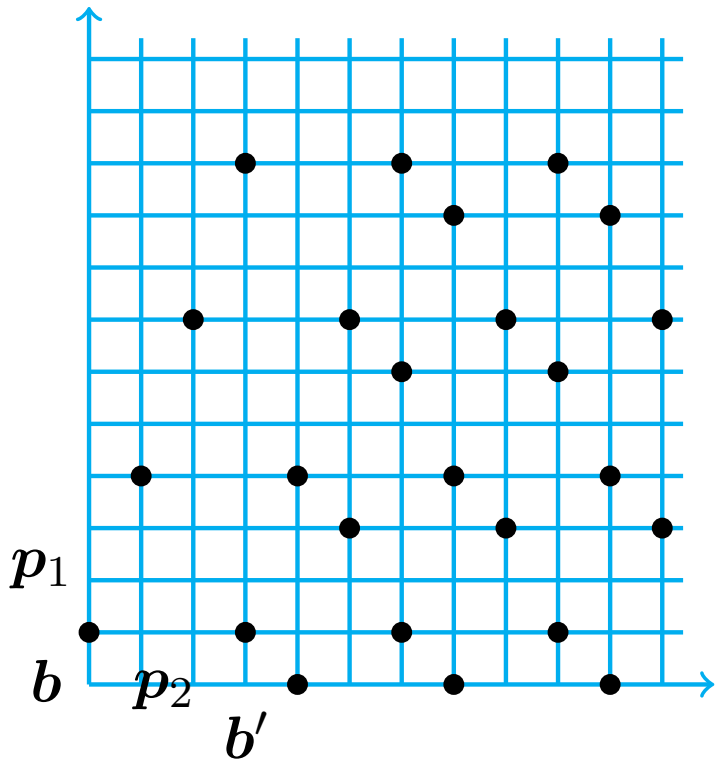
[Parikh (1961)]



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 $L(B, P)$

Sets of solutions to integer programs

Theorem (von zur Gathen and Sieveking, 1978)
For any $S \subseteq \mathbb{Z}^d$, the following are equivalent:

1. $S = \{\boldsymbol{x} \in \mathbb{Z}^d \colon A \cdot \boldsymbol{x} \geq \boldsymbol{c}\}$ for some $A \in \mathbb{Z}^{m \times d}$ and $\boldsymbol{c} \in \mathbb{Z}^m$; and
2. $S = L(C, Q)$ for some finite sets $C \subseteq \mathbb{Z}^d$ and $Q \subseteq \mathbb{Z}^d$.

In both translations $1 \Rightarrow 2$ and $2 \Rightarrow 1$:

- 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

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

- Definition via certificates
- Definition via nondeterministic machines

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

Definition

The complexity class $\mathbf{NP}$ contains all yes/no questions where, for each input x:

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

  whether the witness is valid can be checked efficiently, that is, in time $\mathrm{poly}(\langle x \rangle, \langle y \rangle)$; and

- if the answer is "no", there is no such witness.

Reductions and $\mathbf{NP}$-complete problems

Definition (polynomial-time reduction)
Problem X reduces to problem Y if
there is a polynomial-time algorithm f such that

$$x \text{ is a yes-instance of } X \iff f(x) \text{ a yes-instance of } Y.$$

Reductions and $\mathbf{NP}$-complete problems

Definition (polynomial-time reduction)
Problem X reduces to problem Y if
there is a polynomial-time algorithm f such that

$$x \text{ is a yes-instance of } X \iff f(x) \text{ a yes-instance of } Y.$$

A problem X is $\mathbf{NP}$-hard if all problems in $\mathbf{NP}$ reduce to X.

A probelm X is $\mathbf{NP}$-complete if it is $\mathbf{NP}$-hard and in $\mathbf{NP}$.

Reductions and $\mathbf{NP}$-complete problems

Definition (polynomial-time reduction)
Problem X reduces to problem Y if
there is a polynomial-time algorithm f such that

$$x \text{ is a yes-instance of } X \iff f(x) \text{ a yes-instance of } Y.$$

A problem X is $\mathbf{NP}$-hard if all problems in $\mathbf{NP}$ reduce to X.

A probelm X is $\mathbf{NP}$-complete if it is $\mathbf{NP}$-hard and in $\mathbf{NP}$.

Theorem (Cook, Levin, 1970s)
Boolean satisfiability (SAT) is $\mathbf{NP}$-complete.

Boolean satisfiability (SAT)

Input: Boolean formula φ in conjunctive normal form (CNF)
Output: Does φ has a satisfying assignment?

SAT is $\mathbf{NP}$-complete.

All known algorithms for SAT require time 2^n in the worst case.
Many instances in practice can be solved fast by SAT solvers.

Complexity classes: brief summary

$\mathbf{P}$: polynomial time (efficiently solvable)

$\mathbf{NP}$: nondeterministic polynomial time
(with efficiently verifiable solutions)

Computational complexity of integer programming

Let IP denote the decision problem for integer programming.

Theorem
IP is **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 $\Pi = \{P\}$. `while(true):`

1. Given $\Pi = \{P_1, \ldots, P_k\}$, 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

$$P_j' = P_j \cap \{\boldsymbol{x} \in \mathbb{R}^d \colon x_i \leq \lfloor \zeta \rfloor\} \quad \text{and}$$
$$P_j'' = P_j \cap \{\boldsymbol{x} \in \mathbb{R}^d \colon x_i \geq \lceil \zeta \rceil\}.$$

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'.

$$
\begin{aligned}
& y \to \max \\
\text{subject to} \quad & 2^t \cdot x = (2^t + 1) \cdot y, \\
& 0 \leq x \leq 2^t, \\
& x, y \in \mathbb{Z}
\end{aligned}
$$

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

$$\sum_{i=1}^{m} x_i \to \min$$

$$\text{subject to} \quad \sum_{j\colon i \in S_j} x_j \geq 1, \qquad i = 1, \ldots, n,$$

$$x_i \geq 0, \qquad i = 1, \ldots, m,$$

$$x_i \in \mathbb{Z}, \qquad i = 1, \ldots, m$$

LP relaxation of the integer program

$$\sum_{i=1}^{m} x_i \to \min$$

$$\text{subject to} \quad \sum_{j\colon i \in S_j} x_j \geq 1, \qquad i = 1, \ldots, n,$$

$$x_i \geq 0, \qquad i = 1, \ldots, m,$$

$$x_i \in \mathbb{R}, \qquad i = 1, \ldots, m$$

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

- Linear programming (LP) is in $\mathbf{P}$: ellipsoid method
- Integer programming (IP) is $\mathbf{NP}$-complete
- Geometry of IP: hybrid linear sets
- Algorithm for IP: branch and bound
- Randomized rounding for combinatorial problems

Agenda

Tuesday	Linear programming
Wednesday	Integer programming
Thursday	Decision procedures for arithmetic theories
Friday	Expressive power of arithmetic theories