Approximation Algorithms

Ruben Hoeksma (University of Bremen)

May 21st, 2019

Linear programming formulations

(Integer) Linear programs ((I)LPs)

What do we use (I)LPs for?

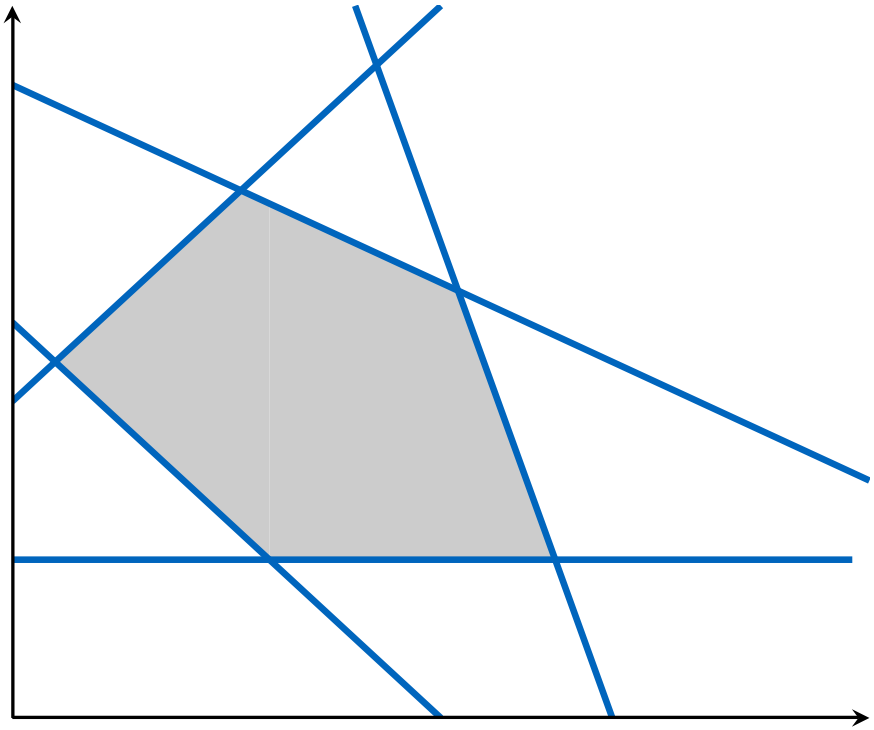
- Formulate optimization problems
- Use solvers
- Get insights in structures of the problem

ILPs are, in general, NP-hard to solve.

Theorem (The ellipsoid method)

LPs are solvable in polynomial time, if a polytime separation algorithm exists.

Polytopes and separation



Definition (Convex polytope / polyhedron)

All points satisfying a system of linear inequalities. Often: polytope bounded; polyhedron unbounded.

Definition (Separation algorithm)

Given a convex polytope/polyhedron P and a vector x, output either:

- $x \in P$, or
- $x \notin P$ and a, b such that $ax > b$ and $ax' \leq b$ for all $x' \in P$.

(Integer) Linear programs ((I)LPs)

What do we use (I)LPs for **in approximation algorithms**?

- Formulate optimization problems
- Relaxations
- Rounding algorithms
- Lower bounds

Note: For the latter, we do not need an exact formulation.

Definition (Integrality gap)

For a given convex polytope $P = \min\{c^T x | Ax \leq b, x \in \mathbb{R}\}$, the integrality gap is

$$\frac{\min\{c^T x | Ax \leq b, x \in \mathbb{N}\}}{\min\{c^T x | Ax \leq b, x \in \mathbb{R}\}}$$

Examples

Definition (Set cover)

Input: a universe $U = \{1, 2, \ldots, n\}$.
Sets $S_1, \ldots, S_m$, such that $S_i \subseteq \{1, \ldots, n\}$.
Each set S_i has associated weight $w_i \geq 0$.

Task: find a cover $C \subseteq \{1, \ldots, m\}$, such that $\bigcup_{j \in C} S_j = U$.

Objective: minimize the total weight $\sum_{j \in C} w_j$.

Definition (Load balancing)

Input: m identical machines,
n jobs with processing times $p_1, \ldots, p_n$.

Task: assign each job $j \in [n]$ to a machine $i \in [m]$.

Objective: minimize the maximum load of any machine
$C_{\max} := \max_{i \in [m]} \sum_{j \,:\, j \to i} p_j$.

Examples

Definition (k-Median)

Input: metric space V with distances d_{ij} for $i, j \in V$ and some $k \in \mathbb{N}$.

Task: find k centers in V, i.e., find $S \subseteq V$ with $|S| \leq k$.

Objective: minimize $\sum_{i \in V} d(i, S)$ with $d(i, S) = \min_{s \in S} d(i, s)$

Definition (k-Center)

Input: metric space V with distances d_{ij} for $i, j \in V$ and some $k \in \mathbb{N}$.

Task: find k centers in V, i.e., find $S \subseteq V$ with $|S| \leq k$.

Objective: minimize $\max_{i \in V} d(i, S)$ with $d(i, S) = \min_{s \in S} d(i, s)$.

Examples

Definition (Knapsack)

Input: a set of n items, N, with values v_i and sizes w_i, and a knapsack size B.

Task: find a subset $S \subset [n]$ such that $\sum_{i \in S} w_i \leq B$.

Objective: maximize $\sum_{i \in S} v_i$.

Definition (Knapsack cover)

Input: a set of n items, N, with costs c_i and sizes w_i, and a knapsack size B.

Task: find a subset $S \subset [n]$ such that $\sum_{i \in S} w_i \geq B$.

Objective: minimize $\sum_{i \in S} c_i$.

Examples

Definition (Uncapacitated facility location)

Input: a set of clients C and a set of facility locations F with metric distances $d_{ij} \in \mathbb{N}$ for all $i, j \in F \cup C$ and facility opening costs $f_i \in \mathbb{N}$ for all $i \in F$.

Task: open a subset S of the facilities, connect all clients.

Objective: minimize $V(S) = \sum_{i \in S} f_i + \sum_{j \in C} d(j, S)$, with $d(j, S) = \min_{i \in S} d_{ij}$.

Definition (Minimum spanning tree)

Input: graph $G = (V \cup \{r\}, E)$, distances $d_e \in \mathbb{N}$.

Task: find a tree T spanning V.

Objective: minimize $\sum_{e \in T} d_e$.