ORF 522

Linear Programming and Convex Analysis

Integer Linear Programming

Marco Cuturi

Today

- Integer programming formulations
  - Interest of integer programming for modeling real-life problems
  - Examples of reformulations
  - Relaxation and strong formulations

Integer Programming Formulations

So far...

- We have often referred to **mathematical programs**:

$$\begin{array}{ll} \text{minimize} & f_0(x) \\ \text{subject to} & f_i(x) \leq 0, \quad i = 1, \ldots, m \\ & h_i(x) = 0, \quad i = 1, \ldots, p \end{array}$$

  where $x \in \boldsymbol{\mathcal{D}} \subset \mathbf{R}^n$.

- For **linear** objectives, **linear** constraints and $\mathcal{D} = \mathbf{R}^n_+$ things have worked well so far:
  - solutions can be computed.
  - simplex, dual simplex, *etc.*
  - ellipsoid method, interior point method, *etc.*

- What if $\mathcal{D}$ is a bit different?

Integer Linear Programs

- What if $\mathcal{D}$ is discrete?
- Some decision variables are **integers**, not fractional numbers:
  - Finance, number of stocks purchased,
  - Number of workers hired for a task,
  - Units of goods ordered/stored at a shop/deposit.
- Sometimes, decision variables are **binary**:
  - have an airplane take/not take off,
  - accept/reject a certain share of applications for a job/grant/journal paper.
- do off-the-shelf algorithms we know always work in such situations?
- **no**, unfortunately.

Integer Linear Programs

- An **integer linear program** is the following program

$$\begin{array}{ll} \text{minimize} & \mathbf{c}^T\mathbf{x} \\ \text{subject to} & A\mathbf{x} = \mathbf{b} \\ & \mathbf{x} \in \mathbb{N}^n \end{array}$$

- A ***mixed* integer linear program**:

$$\begin{array}{ll} \text{minimize} & \mathbf{c}^T\mathbf{x} + \mathbf{d}^{\mathbf{y}} \\ \text{subject to} & A\mathbf{x} + B\mathbf{y} = \mathbf{b} \\ & \mathbf{x}, \mathbf{y} \geq 0, \mathbf{x} \in \mathbb{N}^n \end{array}$$

- A ***binary* or *zero-one* integer program** :

$$\begin{array}{ll} \text{minimize} & \mathbf{c}^T\mathbf{x} \\ \text{subject to} & A\mathbf{x} = \mathbf{b} \\ & \mathbf{x} \in \{0, 1\}^n \end{array}$$

Integer Linear Programs

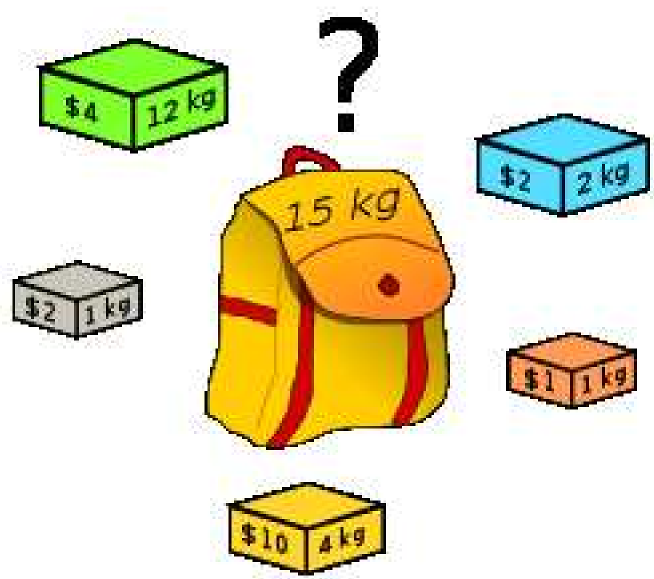
- **pros** of such formulations:
  - by tweaking $\mathcal{D}$, we can incorporate a wide variety of discrete optimizations with such formulations.
  - Indeed, we can considerably enrich the class of problems attacked by LP's.
  - Adding richer conditional constraints.
- **cons**: no universal algorithm.
  - Worse: the resolution of a problem depends heavily on the formulation used.
  - In practice, formulation **matters**.

  - Important difference with standard LP algorithms, where formulations matter less ($e.g.$ canonical and standard formulations, primal & dual)

Let's review some useful formulations

Binary Variables

- Set a variable to 0 or 1: Example: The knapsack problem.
- **Knapsack** (from German knappsack): a bag (as of canvas or nylon) strapped on the back and used for carrying supplies or personal belongings
- Given **objects** with **weights and values** what is the maximal value you can you fit in the bag knowing that it can only accommodate up to a certain weight?



Binary Variables

- This problem is called the (0-1) knapsack problem.
- n items, jth item has value c_j and weight $w_j \Rightarrow$ vectors $\mathbf{c}$ and $\mathbf{w}$.
- Variable $\mathbf{x} \in \{0,1\}^n$ where $x_j = 1$ means the object is in the knapsack.
- A bound K on the maximum weight that can be carried by the knapsack.

$$\begin{array}{ll} \text{maximize} & \mathbf{c}^T\mathbf{x} \\ \text{subject to} & \mathbf{w}^T\mathbf{x} \leq K \\ & \mathbf{x} \in \{0,1\}^n \end{array}$$

Contextual Constraints

- For some practical problems, a decision B can only be taken if another decision A has already been made.
- This can be modelled by adding binary constraints.
  - Let x_A stand for decision A being taken or not, $x_A \in \{0, 1\}$.
  - Let x_B stand for decision B being taken or not, $x_B \in \{0, 1\}$
- if B can only be selected if A is, then this can be naturally formulated:

$$x_B \leq x_A$$

Example: Optimizing Facility Locations

- n potential facility locations to service m existing clients.
- Setting up facility j has a cost of c_j while servicing client i from facility j has a cost of d_{ij}.
- **Problem**: decide which facilities to set-up, while minimizing costs.
  - $\mathbf{y} \in \{0,1\}^n$ defines whether facility j is set up or not through y_j.
  - $\mathbf{x} \in \{0,1\}^{n\times m}$ defines whether client i is services by facility j.

$$\begin{array}{ll} \text{minimize} & \sum_{j=1}^n c_j y_j + \sum_{i=1}^m \sum_{j=1}^n d_{ij} x_{ij} \\ \text{subject to} & \sum_{j=1}^n x_{ij} = 1, \forall i \\ & x_{ij} \leq y_j, \ \forall (i,j) \\ & \mathbf{y} \in \{0,1\}^n, \mathbf{x} \in \{0,1\}^{n\times m} \end{array}$$

  - $x_{ij} \leq y_j$: if no facility j, client i cannot be serviced by j.
  - $\sum_{j=1}^n x_{ij} = 1$: a client i
    - can only be serviced by **at most** one facility **and**
    - needs to be serviced by **at least** one facility.

Disjunctive Constraints

- In some cases it is sufficient that a variable satisfies at least one among possible constraints.
- Example: $\mathbf{a}^T\mathbf{x} \geq b$ **or** $\mathbf{c}^T\mathbf{x} \geq d$ with $\mathbf{a}, \mathbf{c} \geq 0$.
- Modelization: $\mathbf{a}^T\mathbf{x} \geq yb$ **or** $\mathbf{c}^T\mathbf{x} \geq (1-y)d$, $y \in \{0, 1\}$.
- More generally, suppose we are given m constraints $\mathbf{a}_i^T\mathbf{x} \geq b_i$.
- We require that at least k of such constraints are satisfied.
- Can be formulated as:

$$\begin{array}{ll} \text{subject to} & \mathbf{a}_i^T\mathbf{x} \geq y_i b_i, i = 1, \cdots, m \\ & \sum_{i=1}^m y_i \geq k \\ & \mathbf{x} \geq 0, \mathbf{y} \in \{0, 1\}^m \end{array}$$

- second constraint $\Leftrightarrow$ k constraints among m are at least verified.

Restricted Range of Values

- Imagine a variable x is constrained to take values in a subset $\{a_1, a_2, \cdots, a_m\}$
- Turn a discrete problem on integers into a discrete problem on arbitrary values:

$$
\begin{aligned}
&x = \textstyle\sum_{j=1}^m a_j y_j, \\
&\textstyle\sum_{j=1}^m y_j = 1, \\
&y_j \in \{0, 1\}
\end{aligned}
$$

Piecewise Linear Cost Functions

- Suppose $a_1 < a_2 < a_k$ and that a function f is piecewise linear.
- f is defined between a_1 and a_k by the pairs $(a_i,)$
- For any $x \in [a_1, a_k]$, there exists coefficients λ_i such that

$$x = \sum_{i=1}^{k} \lambda_i a_i, \sum_{i=1}^{k} \lambda_i = 1, \lambda_i \geq 0.$$

- This representation is **not unique**.
- It becomes unique if we require that **all but two consecutive** λ_i are **zero**.
- In that case, if $x \in [a_i, a_{i+1}], x$ is uniquely defined as

$$x = \lambda_i a_i + \lambda_{i+1} a_{i+1}, \text{ with } \lambda_i + \lambda_{i+1} = 1, \lambda_i, \lambda_{i+1} \geq 0$$

- We then have for such an x.

$$f(x) = \lambda_i f(a_i) + \lambda_{i+1} f(a_{i+1}) = \sum_{i=1}^{m} \lambda_i f(a_i).$$

Piecewise Linear Cost Functions

- Incorporate the *two consecutive non-zero coefficients* **requirement** $(*)$.
- Let $y_i, i = 1, \cdots, k-1$ be such that $y_i = 1$ iff $a_i \le x \le a_{i+1}$.
- Minimizing f on $[a_1, a_k]$ thus becomes

$$
\begin{array}{ll}
\text{minimize} & \sum_{i=1}^{k} \lambda_i f(a_i) \\[1em]
\text{subject to} & \sum_{i=1}^{k} \lambda_i = 1, \\[1em]
& (*) \begin{cases} \lambda_1 \le y_1, \\ \lambda_i \le y_{i-1} + y_i, i = 2, \cdots, k-1 \\ \lambda_k \le y_{k-1}, \end{cases} \\[1em]
& \sum_{i=1}^{k-1} y_i = 1\,,\ (x \text{ is at most in one interval}) \\[1em]
& \lambda_i \ge 0, i = 1, \cdots, k, \quad y_i \in \{0, 1\}
\end{array}
$$

Relaxations and Formulations

Mathematical Programs: Relaxation

Definition 1. *A mathematical program P' is a* **relaxation** *of P if:*

1. *the feasible region of P' contains the feasible region of P,*

2. *the objective value in P', say $F(x)$, is no worse than that of P, say $f(x)$, for all x in the domain of P. e.g. for minimization, this means $F(x) \geq f(x)$ for all x in the domain of P.*

Integer Program Relaxation

Definition 2. *Given a* mixed *integer linear program,*

$$\begin{array}{ll} \textit{minimize} & \mathbf{c}^T\mathbf{x} + \mathbf{d}^{\mathrm{y}} \\ \textit{subject to} & A\mathbf{x} + B\mathbf{y} = \mathbf{b} \\ & \mathbf{x}, \mathbf{y} \geq 0, \mathbf{x} \in \mathbb{N}^n \end{array}$$

its ***linear programming relaxation*** *is defined as*

$$\begin{array}{ll} \textit{minimize} & \mathbf{c}^T\mathbf{x} + \mathbf{d}^T\mathbf{y} \\ \textit{subject to} & A\mathbf{x} + B\mathbf{y} = \mathbf{b} \\ & \mathbf{x}, \mathbf{y} \geq 0 \end{array}$$

equivalently, the requirement that $\mathbf{x} \in \{0, 1\}^n$ *is usually relaxed to the requirement that each component* x_i *of* $\mathbf{x}$ *is such that* $0 \leq x_i \leq 1$.

Formulations and Relaxations

- Facility Location (FL) Problem:

$$
\begin{array}{ll}
\text{minimize} & \sum_{j=1}^{n} c_j y_j + \sum_{i=1}^{m} \sum_{j=1}^{n} d_{ij} x_{ij} \\
\text{subject to} & \sum_{j=1}^{n} x_{ij} = 1, \forall i \\
& x_{ij} \leq y_j, \forall (i,j) \\
& \mathbf{y} \in \{0,1\}^n, \mathbf{x} \in \{0,1\}^{n \times m}
\end{array}
$$

- An alternative (lighter) formulation: Aggregate Facility Location (AFL):

$$
\begin{array}{ll}
\text{minimize} & \sum_{j=1}^{n} c_j y_j + \sum_{i=1}^{m} \sum_{j=1}^{n} d_{ij} x_{ij} \\
\text{subject to} & \sum_{j=1}^{n} x_{ij} = 1, \forall i \\
& \boldsymbol{\sum_{i=1}^{n} x_{ij}} \leq \boldsymbol{m} y_j, \forall j \\
& \mathbf{y} \in \{0,1\}^n, \mathbf{x} \in \{0,1\}^{n \times m}
\end{array}
$$

- Equivalent formulations but $m+n$ constraints for (AFL), $m+mn$ for (FL).

Relaxation Polyhedrons

- Let us study the relaxations of the equivalent formulations (FL) and (AFL):
- For the original formulation (FL):

$$P_{\mathsf{FL}} = \left\{ (\mathbf{x}, \mathbf{y}) \;\middle|\; \begin{array}{l} \sum_{j=1}^{n} x_{ij} = 1, \forall i \\ x_{ij} \le y_j, \; \forall (i,j) \\ 0 \le x_{ij} \le 1, 0 \le y_j \le 1 \end{array} \right\}.$$

- For its aggregated counterpart (AFL)

$$P_{\mathsf{AFL}} = \left\{ (\mathbf{x}, \mathbf{y}) \;\middle|\; \begin{array}{l} \sum_{j=1}^{n} x_{ij} = 1, \forall i \\ \boldsymbol{\sum_{i=1}^{n} x_{ij}} \le \boldsymbol{m} y_j, \forall j \\ 0 \le x_{ij} \le 1, 0 \le y_j \le 1 \end{array} \right\}.$$

- Interestingly, $P_{\mathsf{FL}} \subset P_{\mathsf{AFL}}$ and this inclusion can be **strict**

Relaxation & Objectives

- The original feasible set L_{IP} is

$$L_{\mathsf{IP}} = \left\{ (\mathbf{x}, \mathbf{y}) \;\middle|\; \begin{array}{l} \sum_{j=1}^{n} x_{ij} = 1, \forall i \\ \boldsymbol{x_{ij}} \leq y_j, \forall j \\ x_{ij}, y_j \in \mathbf{\{0, 1\}} \end{array} \right\}.$$

- Of course, $L_{\mathsf{IP}} \subset P_{\mathsf{FL}} \subset P_{\mathsf{AFL}}$
- If we write
  - z_{IP} for the real optimal value,
  - z_{FL} for the (FL) relaxation,
  - z_{AFL} for the (AFL) relaxation,

  then we naturally have that $z_{\mathsf{IP}} \geq z_{\mathsf{FL}} \geq z_{\mathsf{AFL}}$

  The (AFL) formulation is **lighter** than the FL formulation, but its relaxation provides **a looser lower bound** than (FL) which may be **preferable**.

Relaxation & Objectives

- Let $T = \{\mathbf{x}_1, \cdots, \mathbf{x}_k\}$ be the bounded set of feasible integer solutions of an IP.
- Consider the convex hull of T,

$$\langle T \rangle = \left\{ \sum_{i=1}^{k} \lambda_i \mathbf{x}_i,\ \lambda_i \geq 0, \sum_{i=1}^{k} \lambda_i = 1 \right\}$$

- $\langle T \rangle$ is a polyhedron with **integer extreme points**.
- Any relaxation with feasible set P of an IP defined on T is such that $\langle T \rangle \subset P$.
- Let us imagine a situation where $\langle T \rangle$**=**$\{\mathbf{x} \,|\, D\mathbf{x} \leq \mathbf{d}\}$.
- $\Rightarrow$ Use directly use LP algorithms to optimize on the set $\{\mathbf{x} \,|\, D\mathbf{x} \leq \mathbf{d}\}$.
- We will get an **integer** extreme point in T. The **relaxation is tight**.

Idea: find such a polyhedron $\{\mathbf{x} \,|\, D\mathbf{x} \leq \mathbf{d}\}$ when possible. Usually difficult. Otherwise, prefer a formulation whose relaxation approximates **closely** $\langle T \rangle$.

Relaxation & Objectives

Definition 3. *Consider an IP with feasible solution set T. For two formulations A and B of the same program, whose corresponding LP relaxations have feasible sets P_A and P_B, formulation A is said to be as* ***strong*** *as formulation B if*

$$P_A \subset P_B.$$

- 1st issue: how to find find **strong** formulations?
- 2nd issue: given a strong formulation, how to **compute integer solutions** from a **relaxation**?

Relaxation & Objectives

Definition 4. *Consider an IP with feasible solution set T. For two formulations A and B of the same program, whose corresponding LP relaxations have feasible sets P_A and P_B, formulation A is said to be as **strong** as formulation B if*

$$P_A \subset P_B.$$

- 1st issue: how to find find **strong** formulations?
- 2nd issue: given a strong formulation, how to **compute integer solutions** from a **relaxation**?

Next time

- Practical Methods