IEOR 169: Integer Programming

Igor Molybog

January 27, 2022

Organizational

- Dates, organization, etc. $\rightarrow$ *Syllabus*
- Require accommodations?
  1. Contact me at office hours
  2. Email me at `igormolybog@berkeley.edu` (head with [IEOR 169])
- All information will be uploaded/posted to bCourses
- bCourses Discussion functionality is available for communication. You are welcome to create discussion channels (e.g. Discord, Piazza) and I will gladly share it with everyone

What are optimization problems?

$$\max_{x \in X} f_0(x)$$

Usually, X is given as the set of solutions of a system of inequalities f_i

$$\begin{aligned} \max_{x} \ & f_0(x) \\ \text{s.t.} \ & f_i(x) \leq 0, \quad i = 1, \ldots, m \end{aligned}$$

- $x = (x_1, \ldots, x_n)$: decision variables (**unknowns**)
- $f_0 : \mathbb{R}^n \to \mathbb{R}$: objective function (**data**)
- $f_i : \mathbb{R}^n \to \mathbb{R}, i = 1, \ldots, m$: constraint functions (**data**)
- $f_i \leq 0, i = 1, \ldots, m$: constraints

- $\hat{x}$ is said to be **feasible** if $f_i(\hat{x}) \leq 0, i = 1, \ldots, m$
- $\hat{x}$ is said to be **optimal** if it is feasible and $f_0(\hat{x}) \leq f_0(\tilde{x})$ for any feasible $\tilde{x}$

Why are optimization problems important?

- Saving **resources**: production planning, train scheduling, airline crew scheduling, electricity generation planning
- Increasing **profits**: portfolio optimization, telecommunications
- Saving **lives**: transplant allocation, radiation therapy treatment
- Making **predictions**: data mining & machine learning
- Improving **security**: defense applications
- Theoretical applications: economics, statistics, etc ...

Examples

Data fitting

- variables: model parameters
- constraints: prior information, parameter limits
- objective: measure of misfit or prediction error (opt: plus regularization term)

Examples

Data fitting

- variables: model parameters
- constraints: prior information, parameter limits
- objective: measure of misfit or prediction error (opt: plus regularization term)

Portfolio optimization

- variables: amounts invested in different assets
- constraints: budget, max/min investment per asset, minimum return
- objective: overall risk or return variance

Linear optimization problems

$$
\begin{aligned}
(P): \max_{x} \; & cx \\
\text{s.t.} \; & Ax \leq b \\
& x \geq 0
\end{aligned}
$$

- $c^\top \in \mathbb{R}^n, A \in \mathbb{R}^{m \times n}$ (**data**, **parameters**)
- Can we solve this type of problem efficiently?
  In general, yes (simplex method, interior point method)
- Mature field with advanced tools for practical large-scale applications

Example

Linear programming problem

$$
\begin{aligned}
\max_{x_1,x_2,x_3} \quad & c_1 x_1 + c_2 x_2 + c_3 x_3 \\
\text{s.t.} \quad & a_{11} x_1 + a_{12} x_2 + a_{13} x_3 \leq b_1 \\
& a_{21} x_1 + a_{22} x_2 + a_{23} x_3 \leq b_2 \\
& x_1 \geq 0 \\
& x_2 \geq 0 \\
& x_3 \geq 0
\end{aligned}
$$

- a_{ij}, b_i, c_j are parameters (data)
- x_j are variables

Integer linear optimization problems

(coming back to our main matter of interes . . .)

$$
\begin{aligned}
(P): \max_{x} \; & cx \\
\text{s.t.} \; & Ax \leq b \\
& x \geq 0 \\
& x_i \in \mathbb{Z}, i \in I \subseteq \{1, \ldots, n\}
\end{aligned}
$$

- How has the feasible set changed compared to a linear optimization problem?
- Why do we need **integer variables?**
  Represent logical conditions and countable objects: trains, buses, seats, machines
- Do we know how to solve this type of problem efficiently?
  In general, no . . .

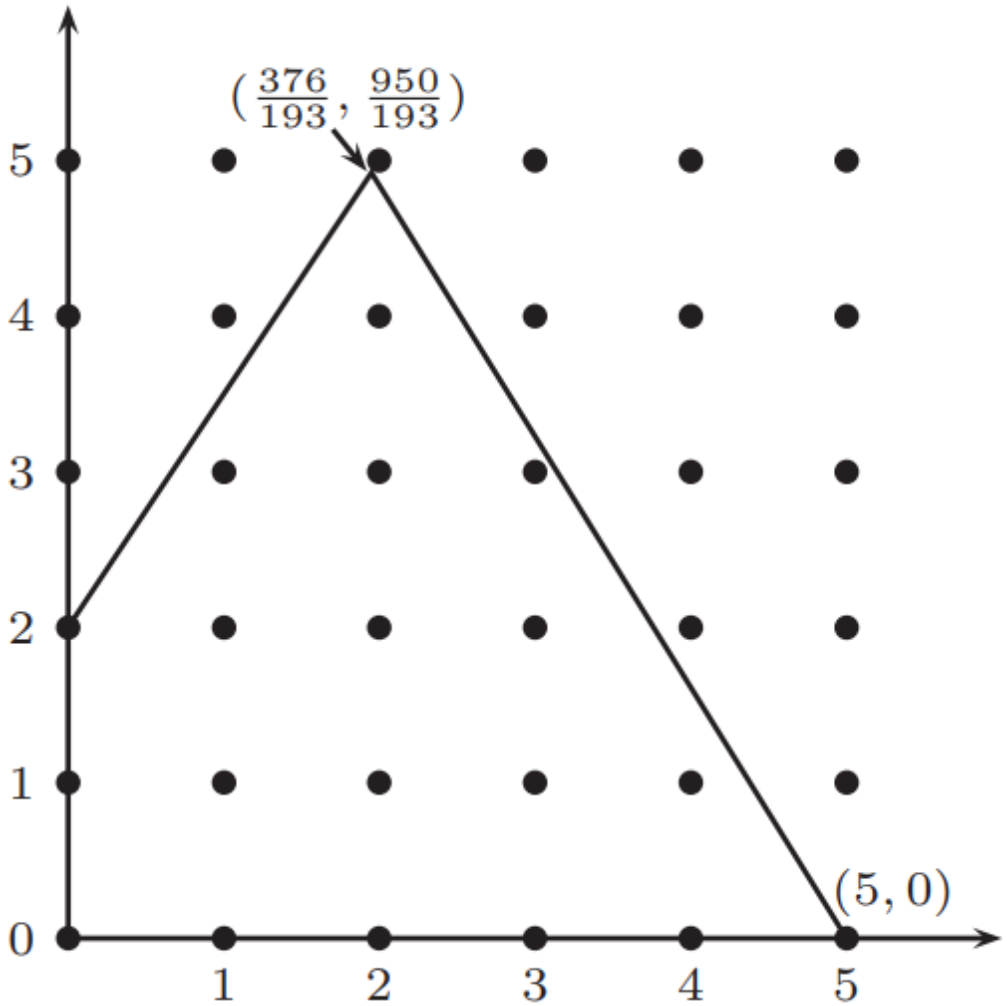
Example of a Linear Integer Program

Example 1.1 Consider the integer program:

$$
\begin{array}{lrrcl}
\max & 1.00x_1 & +0.64x_2 & & \\
 & 50x_1 & +31x_2 & \leq & 250 \\
 & 3x_1 & -2x_2 & \geq & -4 \\
 & x_1, & x_2 & \geq & 0 \text{ and integer.}
\end{array}
$$

What are the feasible points?

Example of a Linear Integer Program



Examples

Assignment problem

- variables: jobs assigned to people
- constraints: one job per person, each job must be done
- objective: cost of executing all jobs

Examples

Assignment problem

- variables: jobs assigned to people
- constraints: one job per person, each job must be done
- objective: cost of executing all jobs

Knapsack problem

- variables: elements in the knapsack
- constraints: knapsack capacity
- objective: total value of elements in the knapsack

Mixed-integer linear programs

$$\begin{aligned} \text{(MIP)} : \max_{x,y} \; & cx + hy \\ \text{s.t.} \; & Ax + Gy \leq b \\ & x \in \mathbb{R}^n_+, y \in \mathbb{Z}^p_+ \end{aligned}$$

- $c^\top \in \mathbb{Q}^n, h^\top \in \mathbb{Q}^p, A \in \mathbb{Q}^{m \times n}, G \in \mathbb{Q}^{m \times p}, b \in \mathbb{Q}^m$ ($\mathbb{Q}$ is the set of rational numbers, **why?**)
- $S := \{x \in \mathbb{R}^n_+, y \in \mathbb{Z}^p_+ \mid Ax + Gy \leq b\}$ is the **feasible set**
- (x, y) is **feasible** if $(x, y) \in S$
- (x^*, y^*) is **optimal** if $cx^* + hy^* \geq cx + hy \; \forall (x, y) \in S$
- $z_{\text{MIP}} := cx^* + hy^*$ is the **optimal value**

Mixed-integer linear programs

- MIP is **infeasible** if $S = \emptyset$
- MIP is **unbounded** if $\forall z \in \mathbb{R}, \exists (x, y) : z < cx + hy$
- Conventions
  - If MIP is infeasible, then $z_{\text{MIP}} = -\infty$
  - If MIP is unbounded, then $z_{\text{MIP}} = +\infty$
- c, h, A, G, b are rational $\implies$ feasible MIP either has an optimal solution or it is unbounded
- Special cases
  - **Integer Program** (IP), only integer variables ($n = 0$)
  - **Binary Program** (BP), only binary variables ($n = 0, y \in \mathbb{B}^p = \{0, 1\}^p$)
  - **Linear Program** (LP), only continuous variables ($p = 0$)

Combinatorial Optimization Problem (COP)

$$\text{(COP)} : \max_{S \subset N} \sum_{j \in S} c_j$$
$$\text{s.t.} \quad S \in \mathcal{F}$$

- $N = \{1, \ldots, n\}$ — finite set of n elements
- $c_j \in \mathbb{R}$: weight (value) of element j.
- $\mathcal{F}$: set of feasible subsets of N.

How is this related to Binary Programs?

What will we learn in this course?

- How to transform real-life problems into mathematical (integer programming) formulations
- What algorithms for their solution exist and how to pick them
- What practical tools to invoke these algorithms are available today

Formulating mixed-integer programs: recipe

1. Define *what appear to be* the **necessary variables**
2. Use these variables to write constraints such that feasible points correspond to feasible solutions
3. Use these variables to define the objective function
4. Complications? Define **new variables** and return to 2.

Example of LP formulation

A medical facility can produce two different vaccines, A and B. One milliliter of vaccine A requires 3 units of saline, 2 units of preservative, and 1 unit of compound A. One milliliter of vaccine B requires 1 unit of saline, 3 units of preservative, and 1 unit of compound B.
The facility has 50 units of saline, 40 units of preservative, 15 units of compound A, and 10 units of compound B.
One milliliter of vaccine A can save 6 lives and one milliliter of vaccine B can save 8 lives.
How many milliliters of each vaccine should the facility produce?

Example of LP formulation

Linear programming problem

$$\begin{aligned} \max_{x,y} \; & 6x + 8y \\ \text{s.t.} \; & 3x + 1y \leq 50 \\ & 2x + 3y \leq 40 \\ & 1x + 0y \leq 15 \\ & 0x + 1y \leq 10 \\ & x \geq 0 \\ & y \geq 0 \end{aligned}$$

Canonical form of LP. What are c, b and A?

Solution of LP example

It turns out (maybe solve this in discussion) that the optimal solution is to make 15 milliliters of vaccine A and 10/3 milliliters of vaccine B. Which will save about 116.66 lives.
but wait.... how many milliliters per vaccine dose? Does it make sense to produce a partial vaccine dose?
If we assume that one dose is one milliliter, then this solution is infeasible. We need another constraint.

Example of IP formulation

Integer (linear) programming problem

$$
\begin{aligned}
\max_{x,y} \quad & 6x + 8y \\
\text{s.t.} \quad & 3x + 1y \leq 50 \\
& 2x + 3y \leq 40 \\
& 1x + 0y \leq 15 \\
& 0x + 1y \leq 10 \\
& x \geq 0 \\
& y \geq 0 \\
& x, y \in \mathbb{Z}
\end{aligned}
$$

Introducing Assignment Problem

You have n special agents available to carry out n missions. Some agents are better for certain missions than others. There is a cost $c_{i,j}$ if agent i carries out mission j. Your goal is to assign the agents to the missions so that the total cost is minimized.

- How to formulate this? What are the variables, constraints, objective function?
- Note $\max_{x \in X} f_0(x) = -\min_{x \in X} -f_0(x)$

Assignment Problem: formulation 1 (poor)

n agents and missions, $c_{i,j}$ if agent i is assigned to mission j, assign agents to missions to minimize total cost.

- Variables?
  - Let x_i be the mission agent i is assigned to
- Constraints?
  - x_i must be an integer from 1 to n (must be a valid mission)
  - $x_i \neq x_k$ for all $i \neq k$ (one agent per job)
- Objective function?
  - Minimize $\sum_{i=1}^{n} c_{i,x_i}$ (total cost)

Assignment Problem: formulation 1 (poor)

$$\begin{aligned} \min_{x_i} \quad & \sum_{i=1}^{n} c_{i,x_i} \\ \text{s.t.} \quad & x_i \leq n \text{ for } i = 1, \ldots, n \\ & x_i \geq 1 \text{ for } i = 1, \ldots, n \\ & x_i \text{ integer for } i = 1, \ldots, n \\ & x_i \neq x_k \text{ for all } i \neq k \end{aligned}$$

Is this an integer linear program? No! ($c_{i,x+y} \neq c_{i,x} + c_{i,y}$)

Assignment Problem: formulation 2 (better)

n agents and missions, $c_{i,j}$ if agent i is assigned to mission j, assign agents to missions to minimize total cost.

- Variables?
  - $x_{ij} = 1$ if agent i is assigned to mission j, and 0 otherwise
- Constraints?
  - agent i does only one job: $\sum_{j=1}^{n} x_{ij} = 1$ for $i = 1, \ldots, n$
  - job j has only one agent: $\sum_{i=1}^{n} x_{ij} = 1$ for $j = 1, \ldots, n$
  - binary variables $x_{ij} \in \{0, 1\}$ for $i = 1, \ldots, n, j = 1, \ldots, n$
- Objective function?
  - Minimize $\sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij}$ (total cost)

Assignment Problem: formulation 2 (better)

$$\min_{x_{ij}} \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij}$$

$$\text{s.t.} \quad \sum_{j=1}^{n} x_{ij} = 1 \text{ for } i = 1, \ldots, n$$

$$\sum_{i=1}^{n} x_{ij} = 1 \text{ for } j = 1, \ldots, n$$

$$x_{ij} \in \{0, 1\} \text{ for } i = 1, \ldots, n \;\; j = 1, \ldots, n$$

Is this an integer linear program? Yes! (Binary Problem)

MaxMin idea

The problem of optimization

$$\max\{\min\{Profit\_S_1, \ldots, Profit\_S_6\}\}$$

Can be re-written as:

$$\begin{aligned} &\max \quad z \\ &\text{s.t. } z \leq Profit\_S_i \qquad \forall i \in \{1, \ldots, 6\} \qquad (1) \end{aligned}$$

General linearization of MaxMin (or MinMax)

Given that $g(y, x)$ is increasing with $y \in \mathbb{R}$ for any $x \in \mathcal{X}$,

$$\max_{x \in \mathcal{X}} \; g(\min\{f_1(x), \ldots, f_m(x)\}, x)$$

is equivalent (in some sense) to

$$\begin{aligned} \max_{x \in \mathcal{X}, z} \quad & g(z, x) \\ \text{s.t. } & z \leq f_i(x) \qquad \forall i \in \{1, \ldots, m\} \end{aligned}$$

Proof by contradiction

General linearization of MaxMin (or MinMax)

Similarly, for $g(y, x)$ increasing with $y \in \mathbb{R}$,

$$\min_{x \in \mathcal{X}} \quad g(\max\{f_1(x), \ldots, f_m(x)\}, x)$$

is equivalent (in some sense) to

$$\begin{aligned} \min_{x \in \mathcal{X}, z} \quad & g(z, x) \\ \text{s.t. } & z \geq f_i(x) \qquad \forall i \in \{1, \ldots, m\} \end{aligned}$$

Note also that

$$\max f(x) = -\min(-f(x))$$

MinMax Example

Suppose we have the problem:

$$\begin{aligned} \min_{x_1, x_2} & \left| x_1 + 5x_2 \right| \\ \text{s.t. } & x_1 - 3x_2 \geq 2 \\ & x_1 \geq 0 \end{aligned}$$

How can we convert it into a Linear Program?

MinMax Example

Suppose we have the problem:

$$\begin{aligned} \min_{x_1,x_2} \; & |x_1 + 5x_2| \\ \text{s.t. } & x_1 - 3x_2 \geq 2 \\ & x_1 \geq 0 \end{aligned}$$

How can we convert it into a Linear Program?
Remember $|x| = \max\{x, -x\}$.

$$\begin{aligned} \min_{x_1,x_2} \; & \max\{x_1 + 5x_2, -x_1 - 5x_2\} \\ \text{s.t. } & x_1 - 3x_2 \geq 2 \\ & x_1 \geq 0 \end{aligned}$$

MinMax Example

Create a new variable z and make:

$$\begin{aligned} \min_{x_1, x_2, z} \quad & z \\ \text{s.t. } & z \geq x_1 + 5x_2 \\ & z \geq -(x_1 + 5x_2) \\ & x_1 - 3x_2 \geq 2 \\ & x_1 \geq 0 \end{aligned}$$

Proof for linearization of MaxMin (or MinMax)

Suppose the solutions are different.

- $g(z_2^\star, x_2^\star) < g(\min\{f_1(x_1^\star), \ldots, f_m(x_1^\star)\}, x_1^\star)$ is not possible because $(z = \min\{f_1(x_1^\star), \ldots, f_m(x_1^\star)\}, x = x_1^\star)$ is feasible for the second problem.
- $g(z_2^\star, x_2^\star) > g(\min\{f_1(x_1^\star), \ldots, f_m(x_1^\star)\}, x_1^\star)$
  By optimality of $x_1^\star$, the inequality holds
  $g(\min\{f_1(x_2^\star), \ldots, f_m(x_2^\star)\}, x_2^\star) \leq g(\min\{f_1(x_1^\star), \ldots, f_m(x_1^\star)\}, x_1^\star)$.
  By definition of the second problem, $z_2^\star \leq \min\{f_1(x_2^\star), \ldots, f_m(x_2^\star)\}$ and from monotonicity of $g(y, x)$ with respect to y we have $g(z_2^\star, x_2^\star) \leq g(\min\{f_1(x_2^\star), \ldots, f_m(x_2^\star)\}, x_2^\star)$ which leads to a contradiction.

Introducing Knapsack Problem

You are planning to break into a vault containing n types of pieces of jewelry. A piece of the j-th type weighs a_j and is worth c_j on the black market. You can carry a total of b pounds. Which pieces should you grab in order to maximize your 'profit'?

- Variables?
  - $x_j = 1$ if item j is selected, and 0 otherwise
- Constraints?
  - weight limit: $\sum_{j=1}^{n} a_j x_j \leq b$
  - integral variables $x_j \in \mathbb{Z}_+$ for $j = 1, \ldots, n$
- Objective function?
  - Maximize $\sum_{i=1}^{n} \sum_{j=1}^{n} c_j x_j$ (profit)

Knapsack Problem: formulation

$$\min_{x_j} \sum_{j=1}^{n} c_j x_j$$

$$\text{s.t.} \sum_{j=1}^{n} a_j x_j \leq b$$

$$x_j \in \mathbb{Z}_+ \text{ for } j = 1, \ldots, n$$

Is this an integer linear program? Yes!

Introducing Set Covering Problem

You are a radical internet anarchist and are looking to establish a p2p mesh network so you don't have to rely on a centralized ISP.
In particular, you have n locations (roofs, telephone poles, etc..) where you have obtained permission to place your network devices. You have identified m different regions in your area that you want to ensure that your devices cover. For each location, you know the cost of setting up the device as well as which regions it will cover. Your goal is to set up the devices in such a way that all m regions have internet access while minimizing the total cost.

Set Covering Problem Formulation

- The set of regions $M = \{1, \ldots, m\}$
- The set of device locations $N = \{1, \ldots, n\}$
- $S_i \subseteq M$ is the set regions covered if you set up a device at location i
- c_i is the cost for setting up a device at location i

Combinatorial optimization problem formulation:

$$\min_{T \subseteq N} \left\{ \sum_{i \in T} c_i \ : \ \cup_{i \in T} S_i = M \right\}$$

Set Covering Problem Formulation

- Variables?
  - $x_i = 1$ if set up location i and $x_i = 0$ otherwise
- Constraints?
  - cover the entire area: $Ax \geq 1$ where A is the incidence matrix ($a_{ij} = 1$ if $i \in S_j$ else 0)
  - binary variables $x_j \in \{0, 1\}$ for $j = 1, \ldots, n$
- Objective function?
  - Minimize $\sum_{i \in N} c_i x_i$ (total cost)

Binary linear programming formulation:

$$\begin{aligned} \min_x \quad & \sum_{i \in N} c_i x_i \\ \text{s.t.} \quad & Ax \geq 1 \\ & x_i \in \{0, 1\} \quad \forall i \in N \end{aligned}$$

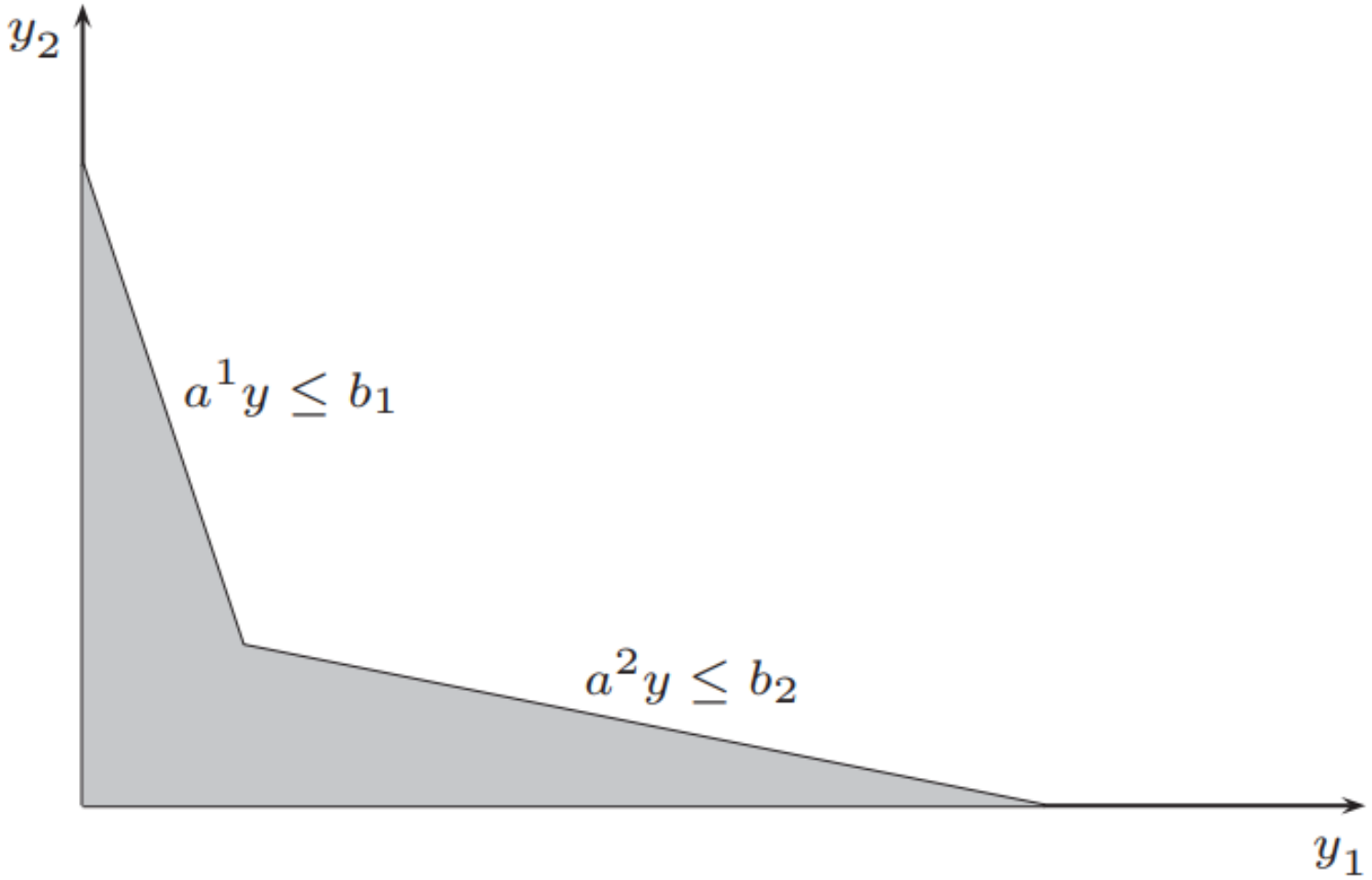
Modeling disjunctions

Suppose we are planning to go shopping and have b_1 dollars in cash and b_2 dollars on a debit card. There are two stores in our town: one accepts only cash and the other accepts only card due to sanitary restrictions. We only have time to visit one of them.

How to model the budgetary restriction of this type?

Modeling disjunctions

Let a^1 and a^2 be the vectors of prices in both stores, and y be the vector of quantities of goods we purchase.



Modeling disjunctions

Let M be a very large constant such that for the optimal y the quantities $a^1 y$ and $a^2 y$ obviously cannot exceed M. Then the following set of constraints model the condition:

$$\begin{aligned} a^1 y - b_1 &\leq M(1 - x_1) \\ a^2 y - b_2 &\leq M(1 - x_2) \\ x_1 + x_2 &= 1 \\ x_1, x_2 &\in \{0, 1\} \end{aligned}$$

Intuition: $x_1 = 1$ if we have decided to visit the first store and $x_2 = 1$ if we have decided to visit the second.

See the file on logical conditions on bCourses to for more.

Formulation

Not all formulations are equal when we start solving them. To formalise it, we need some theory

- Def. 1.1: A **polyhedron** is a subset of $\mathbb{R}^n$ described by a finite number of linear constraints,

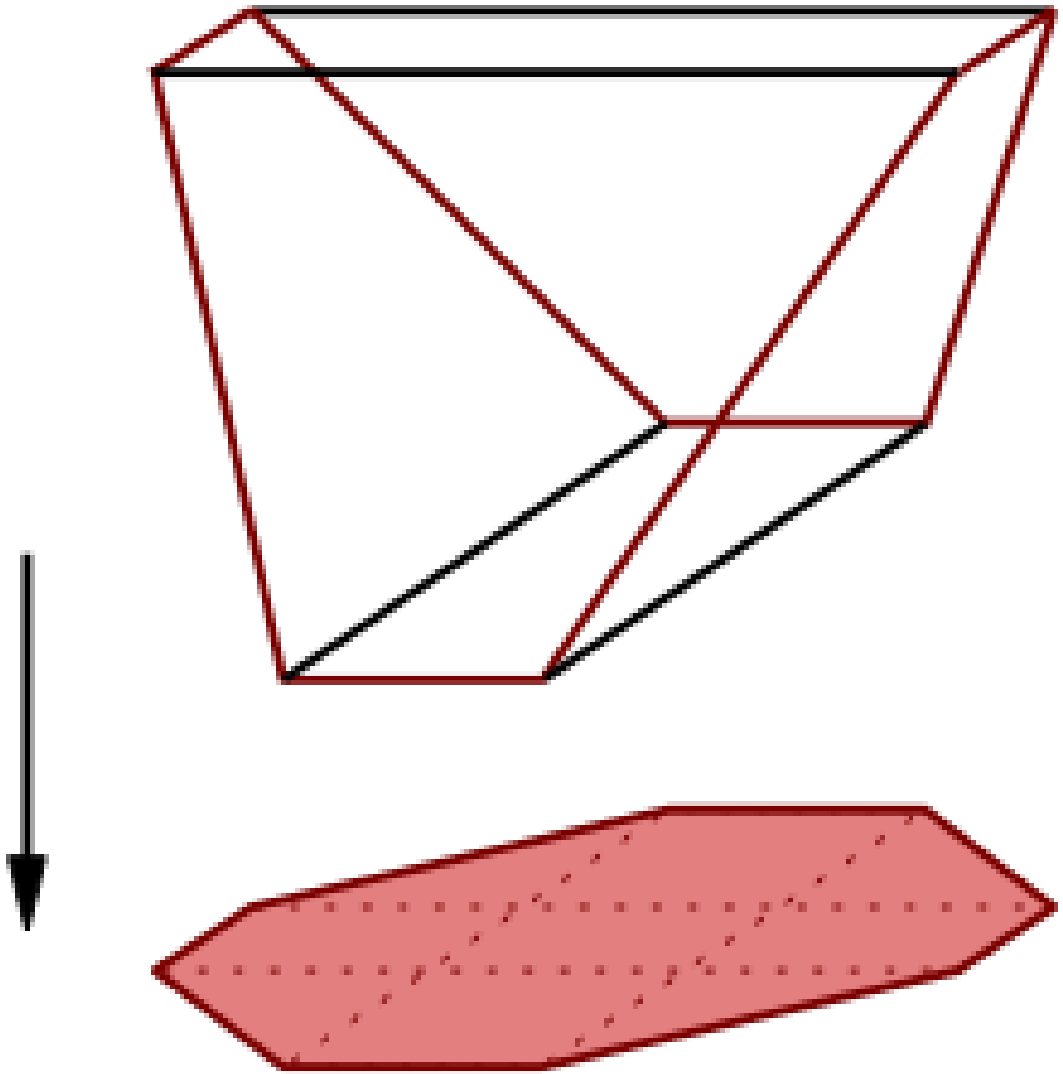
$$P = \{x \in \mathbb{R}^n \mid Ax \leq b\}.$$

- Def. 1.2: A polyhedron is a **formulation** for a set $X \subseteq \mathbb{Z}^n \times \mathbb{R}^p$ if and only if $X = P \cap (\mathbb{Z}^n \times \mathbb{R}^p)$.
- Def 1.6: Given $Q \subseteq \mathbb{R}^n \times \mathbb{R}^p$, the **projection** of Q onto the first n components is

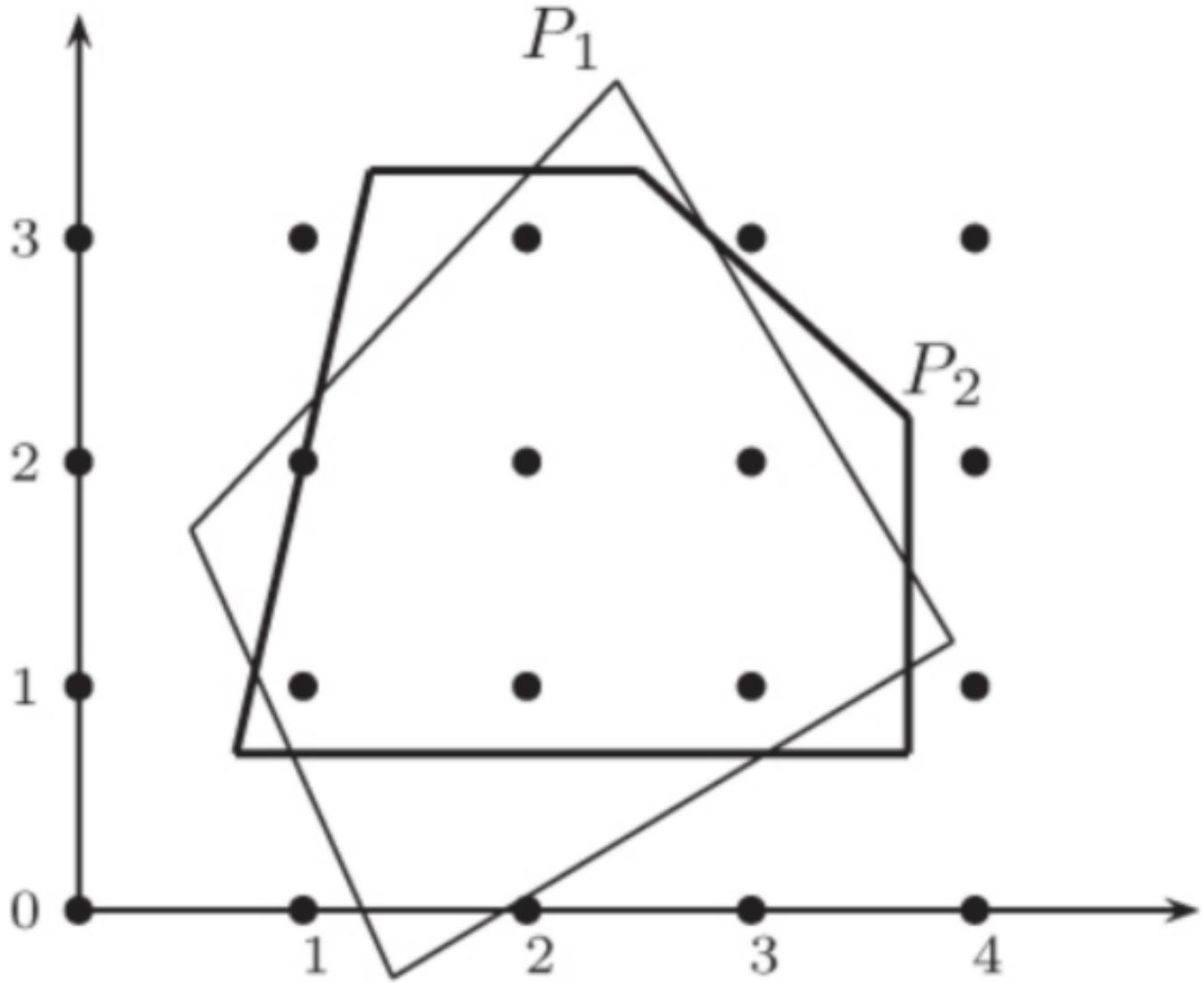
$$\text{proj}_n(Q) = \{x \in \mathbb{R}^n : \exists\, w \in \mathbb{R}^p \ (x, w) \in Q\}$$

- Def: An **extended formulation** for a set $X \subseteq \mathbb{Z}^n \times \mathbb{R}^p$ is a polytope $Q \subseteq \mathbb{Z}^n \times \mathbb{R}^p \times \mathbb{R}^q$ such that $\text{proj}_{n+p} Q$ is a formulation for X.

Projection

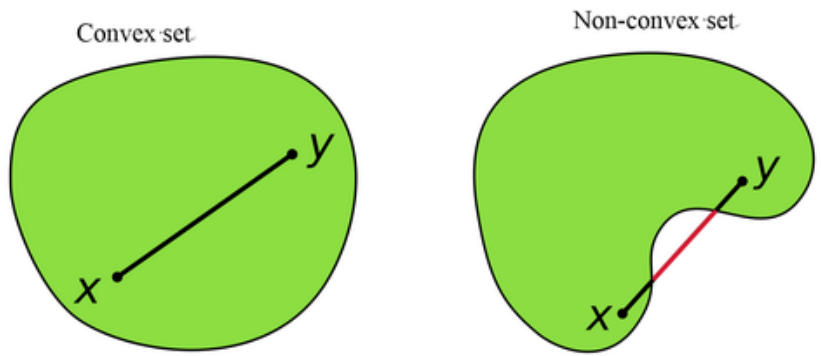


Equivalent formulations



Convex sets

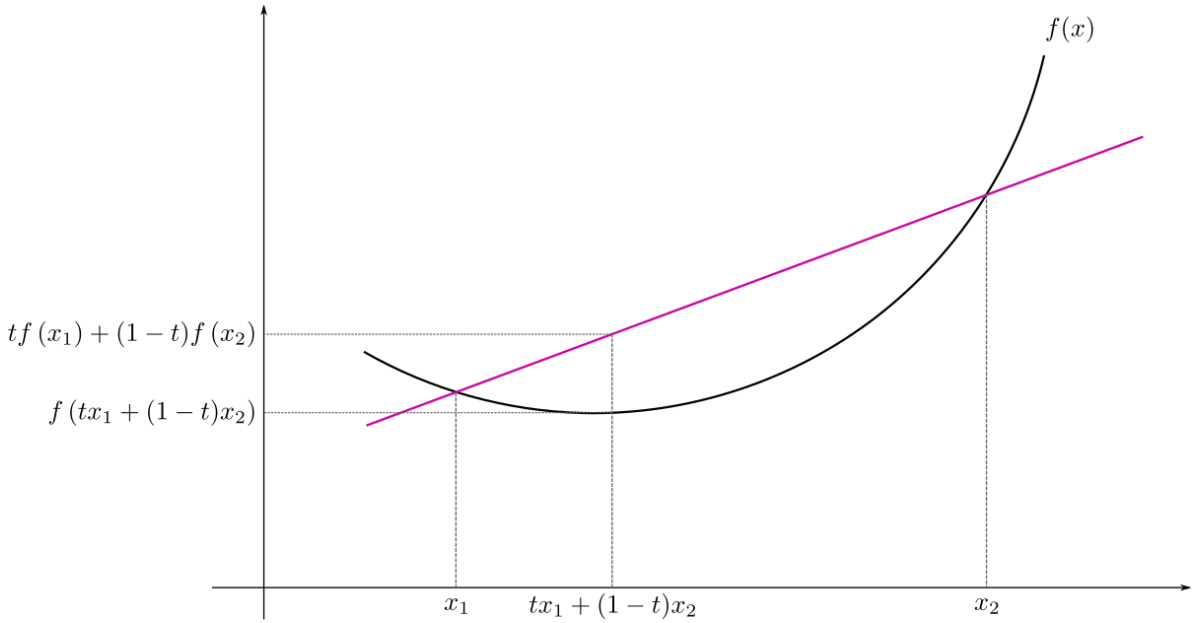
- Def: A linear **combination** of $x^1, \ldots, x^t$ is $y = \sum_{i=1}^{t} \lambda_i x^i$
  - A combination is **conic** if $\lambda_t > 0$
  - A combination is **affine** if $\sum_{i=1}^{t} \lambda_i = 1$
  - A combination is **convex** if it is both conic and affine
- Def. 1.3 A **hull** (conic/affine/convex) of a set X is a set of all possible combinations (conic/affine/convex) of all finite subsets of X. Convex hull is conv(X)
- Def: X is a **convex set** if conv(X) $= X$
- Prop: conv(conv(X)) =conv(X)



https://sahandsaba.com/convex-hull-visualized-using-raphael.html

Convex functions

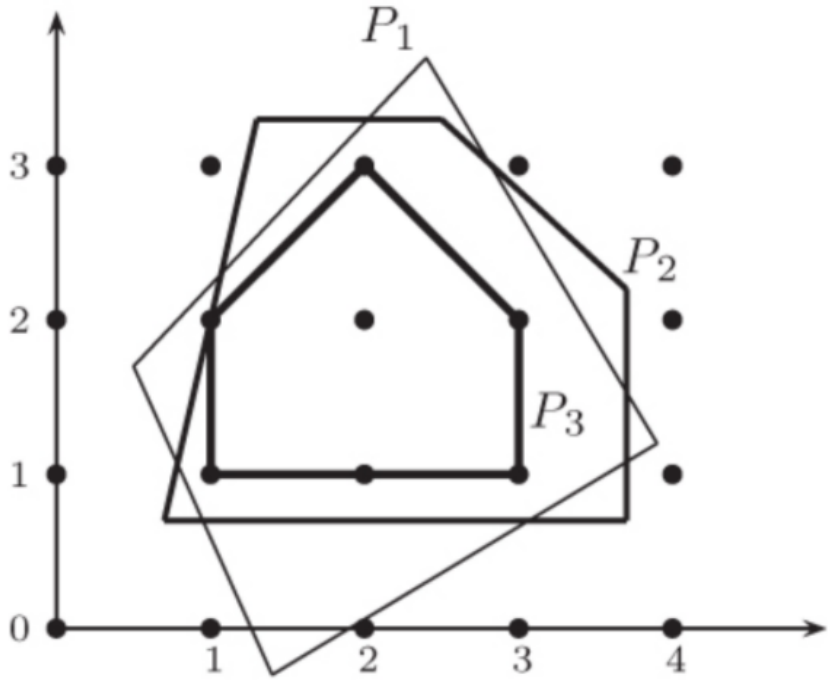
- Def: A function f is **convex** over X if for all $x_1, x_2 \in X$ and $\lambda \in [0, 1]$ it holds $\lambda f(x_1) + (1 - \lambda) f(x_2) \geq f(\lambda x_1 + (1 - \lambda) x_2)$
- Def: A function f is **concave** over X if $-f$ is convex over X.
- Prop: A function is both convex and concave iff it is linear.



Ideal formulation

- Prop: if f is a convex function then $\max_{x \in X} f(x) = \max_{x \in \text{conv}(X)} f(x)$
- Prop. 1.1 If $X = \{x \in \mathbb{R}^n, y \in \mathbb{Z}^p : Ax + Gy \leq b\}$ then conv(X) is a polyhedron

If we have conv(X), we just need to solve a linear problem! Ideal formulation thus describes conv(X)



Comparing formulations

- Def 1.5: Given $X \subseteq \mathbb{Z}^n \times \mathbb{R}^p$ and two formulations P_1 and P_2 for X, we say P_1 is **better** than P_2 if $P_1 \subset P_2$
- Def: If Q_1 and Q_2 are extended formulations, the correspondence between them is determined by the correspondence between their projections.

Example: Uncapacitated Lot Sizing (ULS)

Design a production plan for a single product over an n-period horizon. You are given:

- fixed cost of producing in period t (f_t)
- unit production cost in period t (p_t)
- unit storage cost from t to $t+1$ (h_t)
- demand that must be satisfied in period t (d_t)

ULS: formulation 1

- Variables?
  - $y_t \in \mathbb{R}_+$ is the amount produced in period t
  - $s_t \in \mathbb{R}_+$ is the amount stored from t to $t+1$
  - $x_t = 1$ if production occurs in t and 0 otherwise
- Objective?
  - $\sum_{t=1}^{n} p_t y_t + h_t s_t + f_t x_t$ (cost)
- Constraints?
  - balance on period t

$$s_{t-1} + y_t = d_t + s_t \quad t = 1, \ldots, n$$

  - enforcing definition of x_t

$$y_t \leq M x_t \quad t = 1, \ldots, n$$

ULS: formulation 1

$$\min_{x,s,y} \sum_{t=1}^{n} (p_t y_t + h_t s_t + f_t x_t)$$

$$\text{s.t.} \quad s_{t-1} + x_t = d_t + s_t \quad t = 1, \ldots, n$$

(product is sold or stored)

$$y_t \leq M x_t \quad t = 1, \ldots, n$$

(producing only if $x_t = 1$)

$$s_0 = 0, s_t \geq 0, y_t \geq 0, x_t \in \{0, 1\} \quad t = 1, \ldots, n$$

where $M = \sum_{t=1}^{n} d_t$.

ULS: extended formulation

Trick: introduce additional differentiation. Suppose we distinguish between the products produced today and those produced in the past

- Variables?
  - $w_{it} \in \mathbb{R}_+$ is the amount produced in period i to be consumed in period t ($y_t = \sum_{i=1}^{t} w_{it}$; $s_t = \sum_{i=1}^{t} \sum_{j=t+1}^{n} w_{it}$)
  - x_t as before (indicates production in period t)
- Objective?
  - $\sum_{t=1}^{n} p_t y_t + h_t s_t + f_t x_t$ (same)
- Constraints?
  - balance on period t

$$\sum_{i=1}^{t} w_{it} = d_t \quad t = 1, \ldots, n$$

  - enforcing definition of x_t

$$w_{it} \leq d_t x_i \quad i \leq t = 1, \ldots, n$$

ULS: extended formulation

$$\min_{x,s,y,w} \sum_{t=1}^{n} (p_t y_t + h_t s_t + f_t x_t)$$

$$\text{s.t.} \sum_{i=1}^{t} w_{it} = d_t \quad t = 1, \ldots, n$$

$$w_{it} \leq d_t x_i \quad i \leq t = 1, \ldots, n$$

$$y_t = \sum_{i=1}^{t} w_{it} \quad t = 1, \ldots, n$$

$$s_t = \sum_{i=1}^{t} \sum_{j=t+1}^{n} w_{it} \quad t = 1, \ldots, n$$

$$s_0 = 0, s_t \geq 0, y_t \geq 0, w_{it} \geq 0, x_t \in \{0, 1\}$$

ULS formulations: comparison

Let P_1 correspond to formulation 1 and let Q_2 be the extended formulation. Let $P_2 = \text{proj}(Q_2)$.

- It can be shown that P_2 describes the convex hull of the domain of the problem, thus the extended formulation is the ideal formulation.
- $P_2 \subset P_1$ (the point $y_t = d_t, x_t = d_t/M$ for all t belongs to P_1 but not P_2)

We used target commodities to tighten the formulation. Another way to do that for a problem on graphs is to consider a directed graph instead of an undirected one.

Can we solve optimization problems efficiently?

- First: what do we mean by **solve**?
  - Find a feasible solution $\hat{x} \in X$ such that $f_0(\hat{x}) \geq \sup_{x \in X} f_0(x) - \epsilon$
  - Find a certificate $\hat{y}$ such that the gap between $\sup_{x \in X} f_0(x)$ and $f_0(\hat{x})$ is poly-time verifiable given $\hat{y}$.
- Second: what do we mean by **efficiently**?
  - Essentially, in a polynomial number of arithmetic operations relative to the size of the problem description
- Can all problems be solved efficiently?
  - Most probably, no. There are easy and hard problems.
  - **convex** optimization problems $\overset{usually}{\Longrightarrow}$ easy
  - **non-convex** optimization problems $\overset{usually}{\Longrightarrow}$ hard

Solvers for mixed-integer programming problems

- Software to solve problems in specific formats or passed through specialized libraries
- Commercial (all provide free academic licenses):
  - Gurobi
  - Xpress (FICO)
  - Cplex (IBM)
  - Mosek
  - SCIP (ZIB, open source)
- Free: GLPK (GNU), CBC (Coin-OR)

Algebraic modeling tools

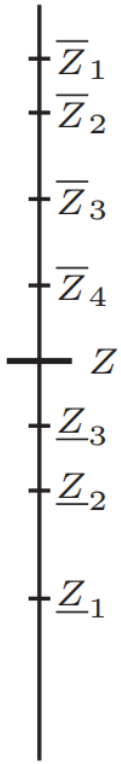
- Software to formulate problems in an algebraic style (e.g. as seen in the class), independent of data, and interact with different solvers using a unique interface
- Commercial (most provide free academic licenses):
  - AMPL
  - GAMS
  - AIMMS
- Free: Pyomo (Python), JuMP (Julia), YalMIP (Matlab)

Proving optimality

Looking for $Z = \max_{x \in X} f(x)$

- Dual bounds (upper bounds $\overline{Z}$ for max):
  - Relaxations:
    - Linear (Section 2.2)
    - Combinatorial (Section 2.3)
    - Lagrangian (Section 2.4, Chapter 10)
    - ...
  - Duality (Section 2.5)
- Optimal value Z
- Primal bounds (lower bounds $\underline{Z}$ for max)
  - Restrictions (Section 2.7)
  - Heuristics (Section 2.7, Chapter 13)
    - Rounding (projection in some metric)
    - Greedy
    - Local search
    - ...

Find $\overline{Z}$ and $\underline{Z}$ such that $\overline{Z} - \underline{Z} < \varepsilon$



Relaxations

- Def. 2.1: A problem of finding $Z^{RP} = \max_{x \in X'} f'(x)$ is a **relaxation** of the problem of finding $Z = \max_{x \in X} f(x)$ if
  - $X \subseteq X'$
  - $f'(x) \geq f(x)$ for all $x \in X$
- Prop 2.1: $Z^{RP} \geq Z$
- Prop 2.3: There are 3 possibilities when solving a relaxation:
  1. We find a solution $x^{RP} \notin X$ or $f(x^{RP}) \neq f'(x^{RP})$: we have obtained a dual bound (upper bound) on Z
  2. We find a solution $x^{RP} \in X$ and $f(x^{RP}) = f'(x^{RP})$: we have solved the original problem
  3. (RP) is infeasible $\implies$ original problem is infeasible

Linear programming relaxations

- Def. 2: Consider the mixed-integer program with a formulation $P = \{x \in \mathbb{R}^{n+p}_{+} \mid Ax \leq b\}$

$$(MIP): \qquad z = \max_x cx$$
$$\text{s.t. } x \in X = P \cap (\mathbb{R}^n \times \mathbb{Z}^p)$$

The **linear programming relaxation** is the linear program

$$(LP): \qquad z = \max_x cx$$
$$\text{s.t. } x \in P$$

- Prop 2.2: For two formulations $P_1 \subset P_2$, the solution values are $z^{LP}_{P_1} \leq z^{LP}_{P_2}$.
- See Example 2.1, Example 2.2 in the book

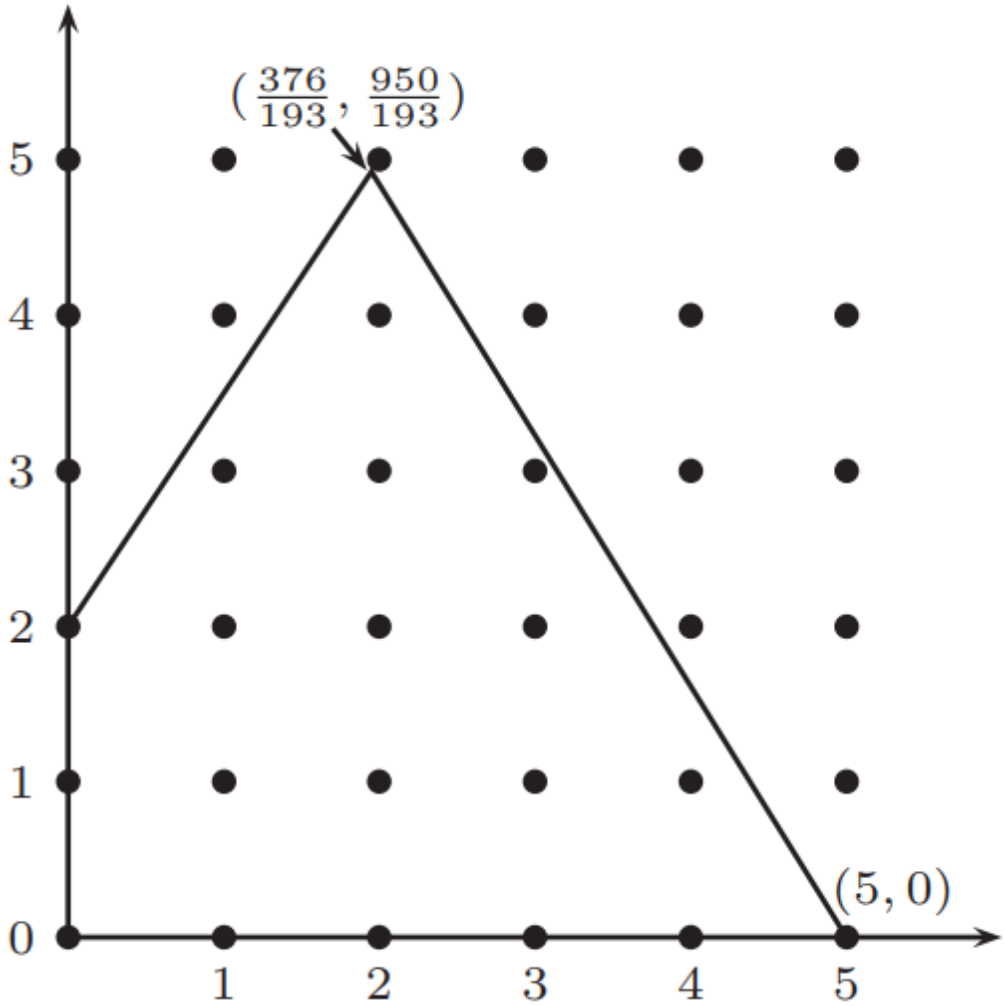
Example of a Linear Integer Program

Example 1.1 Consider the integer program:

$$
\begin{array}{lrcrcl}
\max & 1.00x_1 & +0.64x_2 & & & \\
 & 50x_1 & +31x_2 & \leq & 250 & \\
 & 3x_1 & -2x_2 & \geq & -4 & \\
 & x_1, & x_2 & \geq & 0 \text{ and integer.} &
\end{array}
$$

What happens if we ignore integrity constraint?

Example of a Linear Integer Program



The realaxation might be exact for a different objective.

Combinatorial relaxations

- Def: If the relaxed problem (RP) is a combinatorial optimization problem, it is called a **combinatorial relaxation**
- Dropping constraints in combinatorial optimization may lead to a combinatorial relaxation
- Some combinatorial optimization problems are simpler than the others
- More examples in Section 2.3

Introducing Traveling Salesman Problem (TSP)

There are n cities for a salesperson to visit. The time taken to travel from city i to city j is c_{ij}. Create a travel itinerary so that they visit each location only once and then return to where they start, while minimizing total travel time.
Denote $N = \{1, \ldots, n\}$

- Variables?
  - $x_{ij} = 1$ if the salesperson goes directly from city i to city j, and $x_{ij} = 0$ otherwise (there is no x_{ii})
- Objective?
  - $\sum_{i \in N, j \in N} c_{ij} x_{ij}$ (total time)
- Constraints?
  - salesperson must arrive once at each city: $\sum_{j \in N: j \neq i} x_{ij} = 1 \quad i \in N$
  - salesperson must leave once from each city: $\sum_{i \in N: i \neq j} x_{ij} = 1 \quad j \in N$

Traveling salesman problem: (incomplete) formulation

So far we've got

$$\min_x \sum_{i \in N, j \in N} c_{ij} x_{ij}$$

$$\text{s.t.} \sum_{j \in N: j \neq i} x_{ij} = 1 \quad i \in N$$

(salesperson must arrive once at each city)

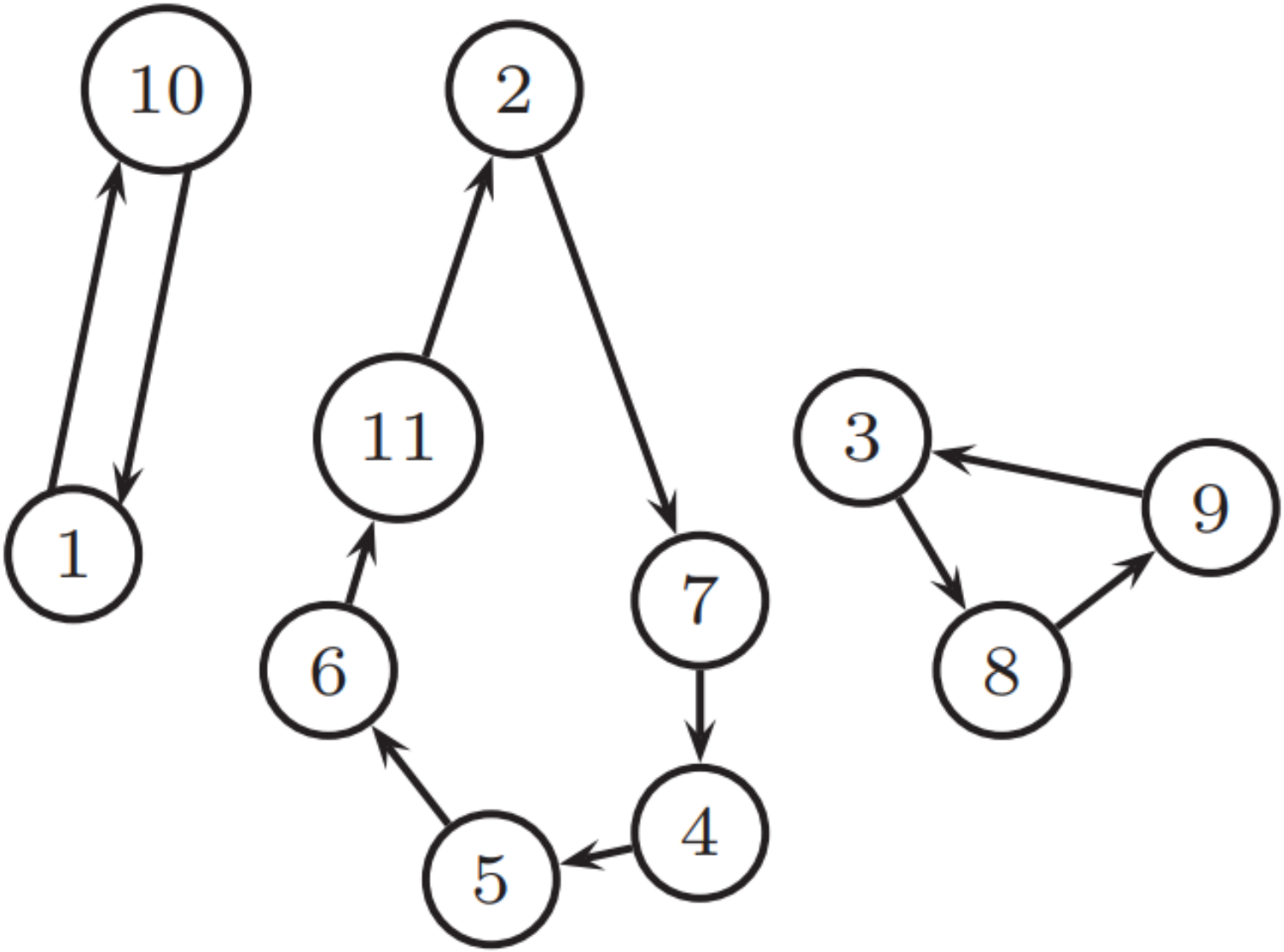
$$\sum_{i \in N: i \neq j} x_{ij} = 1 \quad j \in N$$

(salesperson must leave once from each city)

$$x_{ij} \in \{0, 1\} \quad \forall i \in N, j \in N$$

Looks like the Assignment problem! Is it ok?

Traveling salesman problem: subtours



Traveling salesman problem: subtours

Two alternative conditions to add to the problem formulation to avoid subtours:

- Cut-set constraints: salesperson must leave at least once from each group of cities

$$\sum_{i \in S} \sum_{j \notin S} x_{ij} \geq 1 \quad \forall S \subset N, S \neq \emptyset$$

- Subtour elimination constraints: each group of cities cannot have enough arcs to have a cycle

$$\sum_{i \in S} \sum_{j \in S} x_{ij} \leq |S| - 1 \quad \forall S \subset N, 2 \leq |S| \leq n - 1$$

Traveling salesman problem formulation

$$\min_x \sum_{i \in N, j \in N} c_{ij} x_{ij}$$

$$\text{s.t.} \sum_{j \in N: j \neq i} x_{ij} = 1 \quad i \in N$$

$$\sum_{i \in N: i \neq j} x_{ij} = 1 \quad j \in N$$

$$\sum_{i \in S} \sum_{j \notin S} x_{ij} \geq 1 \quad \forall S \subset N, S \neq \emptyset$$

(salesperson must leave at least once from each group of cities)

$$x_{ij} \in \{0, 1\} \quad \forall i \in N, j \in N$$

Example of a combinatorial relaxation

The incomplete formulation of TSP looked like the assignment problem by a mistake. What if we just drop the subtour condition by design? We get a combinatorial relaxation

$$Z^{TSP} = \min_{T \subseteq A} \left\{ \sum_{(i,j) \in T} c_{ij} \ : \ T \text{ forms a tour} \right\} \geq$$

$$\geq Z^{ASSIGN} = \min_{T \subseteq A} \left\{ \sum_{(i,j) \in T} c_{ij} \ : \ T \text{ forms an assignmnent} \right\}$$

Lagrangian relaxation

- Primal optimization problem

$$\begin{aligned} z = \max_x \; & f_0(x) \\ \text{s.t.} \; & f_i(x) \geq 0 \quad i \in \{1, \ldots, m\} \\ & x \in X \end{aligned}$$

- Let $u \in \mathbb{R}^m_+$, then

$$\begin{aligned} z(u) = \max_x \; & f_0(x) + \sum_{i=1}^{m} u_i f_i(x) \\ \text{s.t.} \; & x \in X \end{aligned}$$

is a **Lagrangian relaxation** of the primal, i.e. $z \leq z(u)$

Lagrangian relaxation for MIP

- Consider the mixed-integer problem

$$\begin{aligned} z = \max_x \; & cx \\ \text{s.t.} \; & Dx \leq d \qquad (P) \\ & x \in X \end{aligned}$$

  where $Dx \leq d$ are m complicating constraints
- Let $u^\top \in \mathbb{R}^m_+$, then

$$\begin{aligned} z(u) = \max_x \; & cx + u(d - Dx) \\ \text{s.t.} \; & x \in X \end{aligned}$$

  is the Lagrangian relaxation of (P)

Lagrangian relaxation example

Original problem:

$$\begin{aligned} \max_x \; & 7x_1 + 8x_2 + 3x_3 \\ \text{s.t.} \; & 2x_2 + 5x_3 \leq 32 \\ & -x_1 + 7x_2 \leq 107 \\ & x \in \mathbb{Z}^3_+ \end{aligned}$$

Lagrangian relaxation:

$$\begin{aligned} \max_x \; & 7x_1 + 8x_2 + 3x_3 + u_1(32 - 2x_2 + 5x_3) + u_2(107 + x_1 - 7x_2) \\ \text{s.t.} \; & x \in \mathbb{Z}^3_+ \end{aligned}$$

Lagrangian relaxation strength

- Th 10.1: Solving the Lagrangian dual for MIP provides the same bound as solving

$$\begin{aligned} z^{LD} = \max_{x} \; & cx \\ \text{s.t.} \; & Dx \leq d \\ & x \in \text{conv}\,(X) \end{aligned}$$

- Lagrangian dual cannot be weaker than the linear relaxation (linear relaxation is a special case of the LD)
- Lagrangian dual of an LP is the dual LP

Duality

- Def. The two problems

$$(P) Z = \max\{c(x) \ : \ x \in X\}$$
$$(D) W = \min\{\omega(u) \ : \ u \in U\}$$

  form a **weak dual pair** if $c(x) \leq \omega(u)$ for all $x \in X$ and $u \in U$ and a **strong dual pair** if $Z = W$.
- Def: x are the **primal variables**, u are the **dual variables**
- Def: $|W - Z|$ is the **duality gap**

Linear dual

- Prop. Dual of a relaxation is a dual for the original problem.

Example:

$$Z = \max\{xa \ : \ Ax \leq b, x \in \mathbb{Z}_+^n\}$$
$$Z^{LP} = \max\{xa \ : \ Ax \leq b, x \in \mathbb{R}_+^n\}$$
$$W^{LP} = \min\{ub \ : \ uA \geq c, u \in \mathbb{R}_+^m\}$$

Lagrangian dual

- Def: We can obtain the best upper bound $z(u)$ on z by solving

$$z^{LD} = \min_{u} z(u)$$
$$\text{s.t. } u \geq 0,$$

which is known as the **Lagrangian dual problem**

Example:

$$\min_{u}\{\max_{x \in \mathbb{Z}^3_+} 7x_1 + 8x_2 + 3x_3 + u_1(32 - 2x_2 + 5x_3) + u_2(107 + x_1 - 7x_2)\}$$
$$\text{s.t. } u \in \mathbb{R}^3_+$$

Solving the Lagrangian dual problem

- The Lagrangian dual is a convex problem ($z(u)$ is a convex function)
- We can use first-order methods to solve the Lagrangian dual problem to global optimality
- Bad news: $z(u)$ might be non-differentiable (more on this in Chapter 10)
- Lagrangian dual is only a weakly dual unless special conditions are satisfied (e.g. Slater's conditions)

References: Duality

Duality for LP

- Wiki: LP Dual
- The Dual LP and Approx Algos
- Duality (with generic examples)

Duality in general

- Wiki: Duality
- Duality Theory of Constrained Optimization

The challenge of primal bounds

- The only way to obtain a primal bound is to find a feasible point
- The problem of feasibility for MIP is as hard as the problem of optimization (by the binary search argument)
- Practitioners are using heuristics to find the best primal bounds

Restrictions

- Def. 2.8: A problem of finding $Z^{RE} = \max_{x \in X'} f'(x)$ is a **restriction** of the problem of finding $Z = \max_{x \in X} f(x)$ if
  - $X' \subseteq X$
  - $f'(x) \leq f(x)$ for all $x \in X$
- Prop 2.1: $Z^{RE} \leq Z$

Example:

Original problem:

$$\begin{aligned} \max_{x,s,y} \; & b^\top x + c^\top y \\ \text{s.t.} \; & y \leq Ax \\ & y \in \mathbb{R}^n_+ \\ & x \in X \subset \mathbb{Z}^m_+ \end{aligned}$$

Restriction:

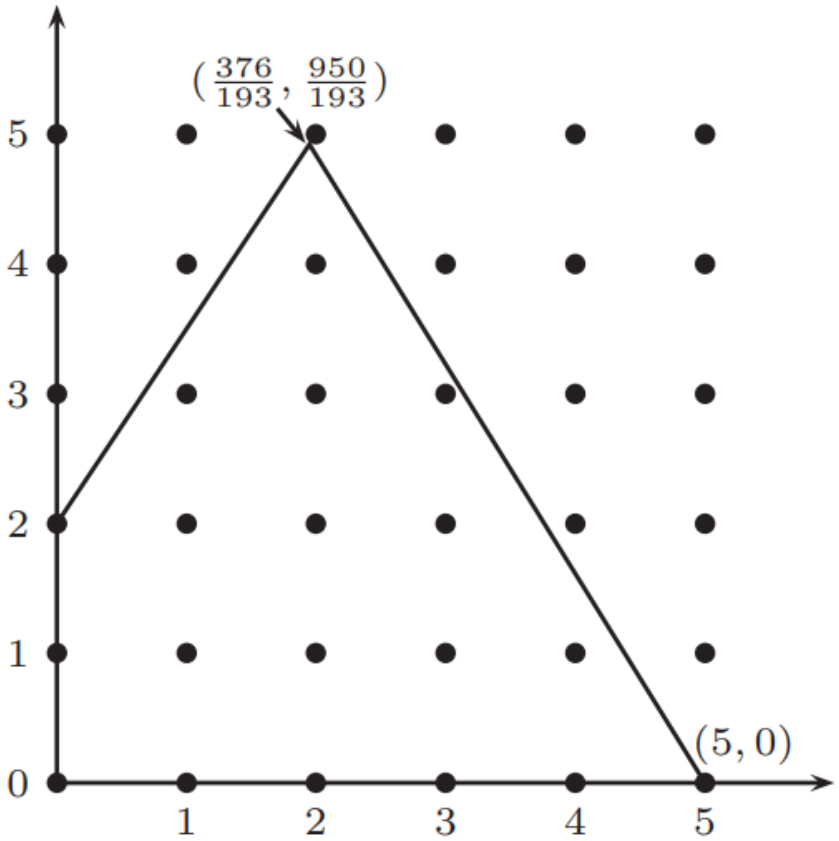
$$\begin{aligned} \max_{x,s,y} \; & b^\top x + c^\top y \\ \text{s.t.} \; & y = Ax \\ & y \in \mathbb{R}^n_+ \\ & x \in X \subset \mathbb{Z}^m_+ \end{aligned}$$

Restriction revised:

$$\begin{aligned} \max_{x,s,y} \; & (b^\top + c^\top A)x \\ \text{s.t.} \; & x \in X \end{aligned}$$

Heuristic: rounding

- LPs can be solved efficiently $\rightarrow$ naive strategy for MIP:
  1. Form linear relaxation (drop integrality constraints)
  2. Solve the LP
  3. Round the components of the LP solution



Heuristic: greedy

Given a combinatorial problem $\min_{S \subset N}\{c(S) \ : \ v(S) \geq k\}$, a greedy heuristic is a scheme:

1. Find $\arg\min_{j \in N} \frac{c(S \cup \{j\}) - c(S)}{v(S \cup \{j\}) - v(S)}$
2. $N \leftarrow N \backslash \{j\}$. If $v(S \cup \{j\}) \geq k$ set $S \leftarrow S \cup \{j\}$

Which iterates until N becomes empty

Given a set function $f : 2^N \rightarrow \mathbb{R}$ and an IP of the form $\max_{x \in \mathbb{Z}^n_+}\{cx \ : \ \sum_{j \in S} x_j \leq f(S) \text{ for } S \subseteq N\}$, a greedy heuristic is a scheme:

- Order $c_1 \geq \ldots c_r > 0 \geq c_{r+1} \geq c_n$
- Set $x_i = f(\{1, \ldots i\}) - f(\{1, \ldots i-1\})$ if $i \leq r$ and $x_i = 0$ otherwise

Greedy heuristic example: Binary knapsack

$$\begin{aligned} \max_{x,s,y} \; & 12x_1 + 8x_2 + 17x_3 + 11x_4 + 6x_5 + 2x_6 + 2x_7 \\ \text{s.t.} \; & 4x_1 + 3x_2 + 7x_3 + 5x_4 + 3x_5 + 2x_6 + 3x_7 \leq 9 \\ & x \in \{0,1\}^7 \end{aligned}$$

- Ordering: $\frac{c_j}{a_j} \geq \frac{c_{j+1}}{a_{j+1}}$
- Following the greedy scheme, we set $x_1 = 1$, $x_2 = 1$, skip $x_3 = x_4 = x_5 = 0$ as they cannot fit in, take $x_6 = 1$ and skip $x_7 = 0$.
- $x^G = (1, 1, 0, 0, 0, 1, 0)$ with value $Z^G = 22$

Heuristic: local search

Given a combinatorial problem $\min_{S \subset N}\{c(S) \ : \ g(S) = 0\}$ and defined

- Local neighborhood $Q(S) \subset 2^{\{1,\ldots,n\}}$
- Goal function $f(S) = c(S) + \alpha g(S)$

The local search heuristic is a scheme

- $S \leftarrow \arg\min_{S' \in Q(S)} f(S')$

Which iterates until S is locally optimal

Local search example: Graph Equipartition Problem

We are managing a communication network represented with a graph $G = (V, E)$ that needs security improvements ($|V| = n$). Our adversary can put a number of our communication lines (E) out of service, and the greatest damage is done if our network is being split into two equal unconnected parts. How can we know where to build new lines to prevent that?
Introduce a cut set $\delta(S, V \backslash S) = \{(i, j) \in E \ : \ i \in S, j \in V \backslash S\}$

$$\min_{|S|=n/2} |\delta(S, V \backslash S)|$$

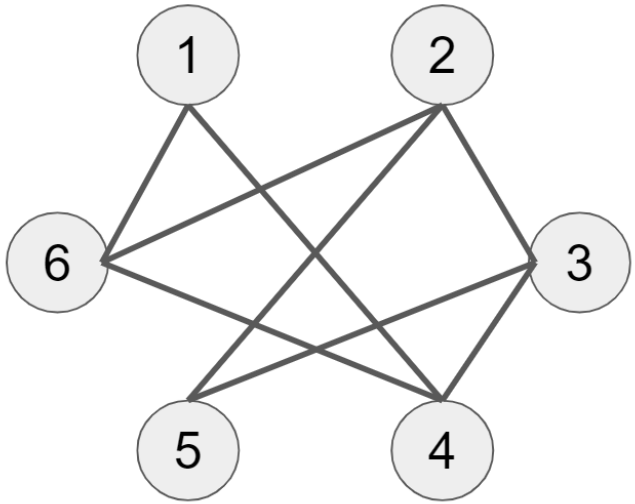
We will need to build lines between S and $V \backslash S$

Local search example: Graph Equipartition Problem

$$\min_{S \subset V} |\delta(S, V \backslash S)|$$

Define neighborhood: $Q(S) = \{T \subset V \ : \ |T \backslash S| = |S \backslash T| = 1\}$

- $S^0 = (1, 2, 3)$; $|\delta(S^0, V \backslash S^0)| = 6$; $Q(S^0) = \{(1, 2, 4), (1, 2, 5), (1, 2, 6), (1, 3, 4), (1, 3, 5), (1, 3, 6), (2, 3, 4), (2, 3, 5), (2, 3, 6)\}$ with costs $6, 5, 4, 4, 5, 6, 5, 2, 5$
- $S^1 = \{2, 3, 5\}$ with cost 2. $Q(S^1)$ does not have a better solution.



Heuristic: Simulated Annealing

Given a combinatorial problem $\min_{S \subset N}\{c(S) \ : \ g(S) = 0\}$ and defined

- Local distribution $\mathbb{P}_S(S')$ supported on a neighborhood $Q(S) \subset 2^{\{1,\ldots,n\}}$
- Goal function $f(S) = c(S) + \alpha g(S)$

The simulated annealing is a scheme

1. Iterate L_1 times
   1. Pick S' from $\mathbb{P}_S(S)$
   2. Let $\Delta \leftarrow f(S') - f(S)$
   3. If $\Delta \leq 0$ set $S \leftarrow S'$
   4. If $\Delta > 0$, set $S \leftarrow S'$ w.p. $e^{-\Delta/T}$
2. Reduce T

Which iterates L_2 times

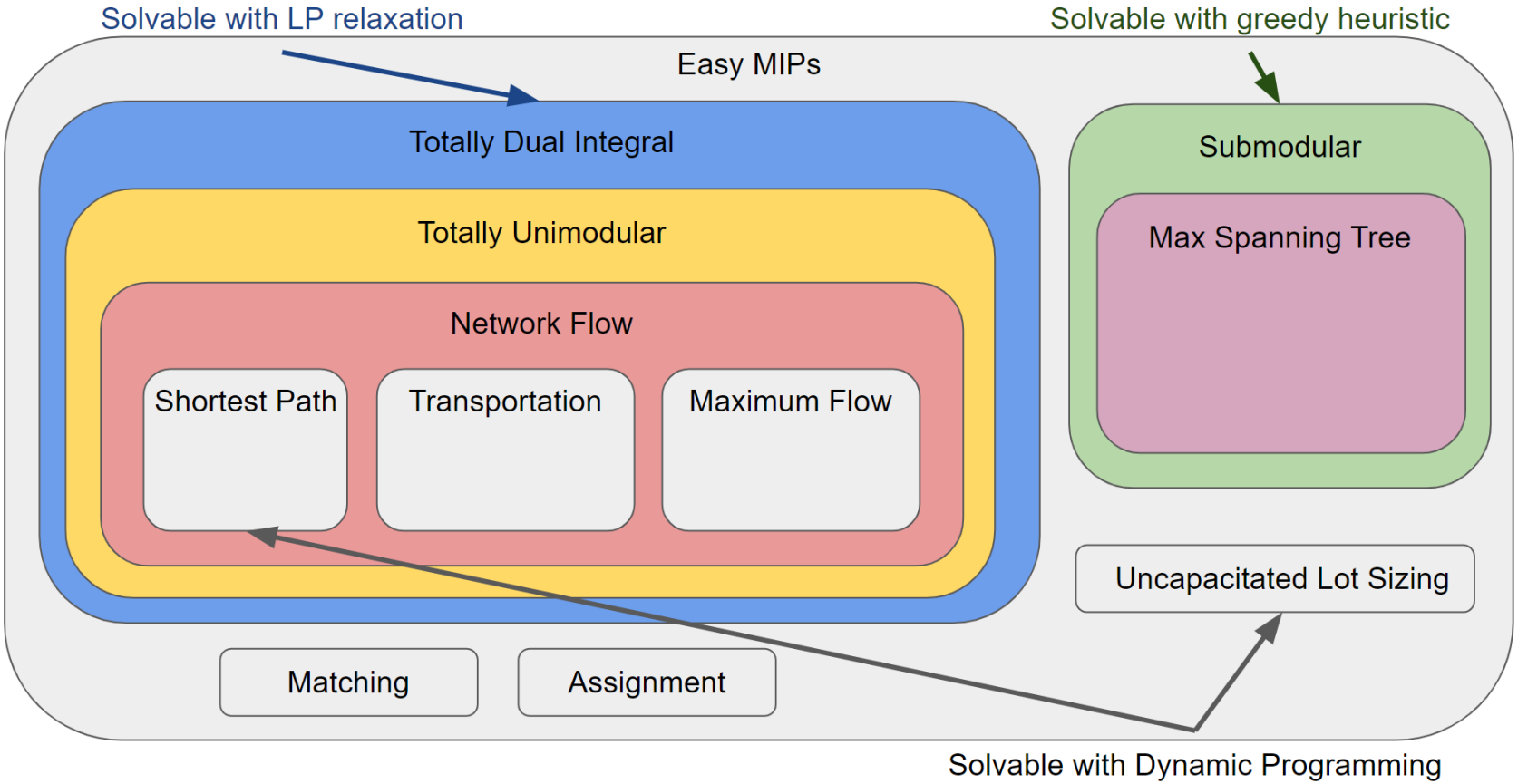
Problems and classes of problems

$$\max\{\eta \ : \ \eta \in X\}$$

- If you are given a particular set X (in the form of decimal numbers), it is an **instance of a problem**
- If you are given a family $\mathcal{X}$ of sets to which X can belong to, it is a **problem**
  - Example: X is given through any finite set of linear inequalities with rational coefficients (Linear Programming problem)
- If you are given a class $\mathcal{C}$ to which the family $\mathcal{X}$ of sets X can belong to, it is a **class of problems**
  - Example: the set of all problems for which there exists an algorithm that solves its instance in a time, polynomial in the size of the instance (Polynomial class)

Map of easy problems

Easy problems: those guaranteed to be solved in polynomial time by a known algorithm



What do we need for the LP relaxation to work?

$$(IP)\ \min_{x \in X} cx \quad vs. \quad (LP)\ \min_{x \in P} cx; \quad X = P \cap \mathbb{Z}^n; \quad P = \{x \in \mathbb{R}^n_+ \ : \ Ax = b\}$$

- What do we know about the solution of (LP)? (assuming A is full row rank)
- From linear programming theory, we know that $x^* = (x_B, x_{NB})$ (basic, non-basic variables) and

$$x_B = B^{-1}b \qquad x_{NB} = 0$$

  Where B is a sub-matrix of A.
- Cramer's rule: $B^{-1} = \mathrm{adj}(B)/\det(B)$.
- Obs 3.1: If $A \in \mathbb{Z}^{m \times n}$, $b \in \mathbb{Z}^m$ and $\det(B) = \pm 1$ then $x_B \in \mathbb{Z}^n$!!! (sufficient condition)

Total unimodularity

- Def 3.2: A matrix A is **totally unimodular** (TU) if every square submatrix of A has determinant $+1$, -1 or 0.
- Prop 3.1: A matrix A is TU if and only if
  1. $A^\top$ is TU
  2. (A, I), where I is the identity matrix, is TU
- Prop 3.2: A matrix is TU if all of the following conditions are satisfied:
  1. $a_{ij} \in \{-1, 0, 1\}$ for all i, j
  2. Each column contains at most two non-zero coefficients
  3. There exists a partition (M_1, M_2) of the set M of rows, such that each column j containing two non-zero coefficients satisfies $\sum_{i \in M_1} a_{ij} - \sum_{i \in M_2} a_{ij} = 0$.

$$\begin{bmatrix} 1 & -1 & -1 & 0 \\ -1 & 0 & 0 & 1 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Role of Total Unimodularity

- Prop. 3.3: The problem of the form $\max_{x \in X} cx$ with $X \in \mathcal{X}$ where $\mathcal{X}$ is a family of sets (with arbitrary n and m)

$$\left\{ \{ x \in \mathbb{Z}_+^n \ : \ Ax \leq b \} \ : \ A \in \mathbb{Z}^{m \times n} \text{ is TU}, b \in \mathbb{Z}^m \right\}$$

  is solvable with a linear programming relaxation
- The statement above only sets conditions on A, while b can be any integral vector. A generalization requires a condition on both A and b simultaniously, called Total Dual Integrality (See Def 3.7 and Th 3.8)
- The requirement of TU is necessary. If A is not TU, there are vectors b and c for which $Z^{IP} \neq Z^{LP}$

Examples of problems with TU matrices/submatrices

- Assignment problem
- Minimum cost network flows
  - Transportation
  - Maximum flow
  - Shortest path

Example: Minimum cost network flow

- Directed graph $D = (V, A)$,
- Arc capacities h_{ij}, for $(i, j) \in A$
- Arc unit flow costs c_{ij}, for $(i, j) \in A$
- (optional) Demands b_i, for $i \in V$

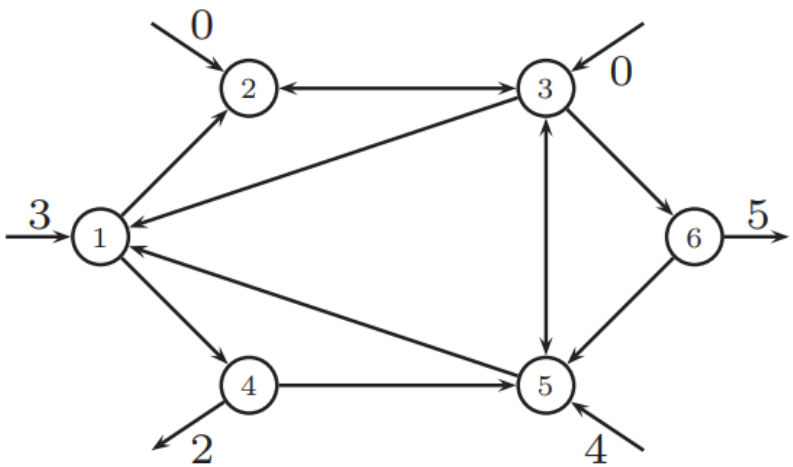
Example of TU problem: Minimum cost network flow

Let $V^+(i) = \{k \ : \ (i,k) \in A\}$ and $V^i(i) = \{k \ : \ (k,i) \in A\}$

$$\min_{x \in \mathbb{Z}_+^{|A|}} \sum_{(i,j) \in A} c_{ij} x_{ij}$$

$$\text{s.t.} \quad \sum_{k \in V^+(i)} x_{ik} - \sum_{k \in V^-(i)} x_{ki} y \leq Ax \quad i \in V$$

$$0 \leq x_{ij} \leq h_{ij} \quad (i,j) \in A$$



Example of TU problem: Minimum cost network flow

The balance equations for the graph:

x_{12}	x_{14}	x_{23}	x_{31}	x_{32}	x_{35}	x_{36}	x_{45}	x_{51}	x_{53}	x_{65}		
1	1	0	−1	0	0	0	0	−1	0	0	=	3
−1	0	1	0	−1	0	0	0	0	0	0	=	0
0	0	−1	1	1	1	1	0	0	−1	0	=	0
0	−1	0	0	0	0	0	1	0	0	0	=	−2
0	0	0	0	0	−1	0	−1	1	1	−1	=	4
0	0	0	0	0	0	−1	0	0	0	1	=	−5

The additional constraints are $0 \leq x_{ij} \leq h_i j$

- Prop 3.4: The constraint matrix is TU.

Minimum Cost Flow Model Example

A company manufacturing chairs has four plants located around the country. The cost of manufacture, excluding raw material, per chair and the minimum and maximum monthly production for each plant is shown in the following table.

		Production	
Plant	Cost per chair	Maximum	Minimum
1	\$5	500	0
2	\$7	750	400
3	\$3	1000	500
4	\$4	250	250

Twenty pounds of wood is required to make each chair. The company obtains the wood from two sources. The sources can supply any amount to the company, but contracts specify that the company must buy at least eight tons of wood from each supplier. The cost of obtaining the wood at the sources is:

- Source 1 = \$0.10/pound
- Source 2 = \$0.075/pound

Shipping cost per pound of wood between each source and each plant is shown in cents by the following matrix:

		Plant			
		1	2	3	4
Wood	1	1	2	4	4
Source	2	4	3	2	2

Minimum Cost Flow Model Example

The chairs are sold in four major cities: New York, Chicago, San Francisco and Austin. Transportation costs between the plants and the cities are shown in the following matrix: (All costs are in dollars per chair.)

		Cities			
		NY	A	SF	C
Plant	1	1	1	2	0
	2	3	6	7	3
	3	3	1	5	3
	4	8	2	1	4

The maximum and minimum demand and the selling price for the chairs in each city is shown in the following table:

		Demand	
City	Selling price	Maximum	Minimum
New York	\$20	2000	500
Austin	\$15	400	100
San Francisco	\$20	1500	500
Chicago	\$18	1500	500

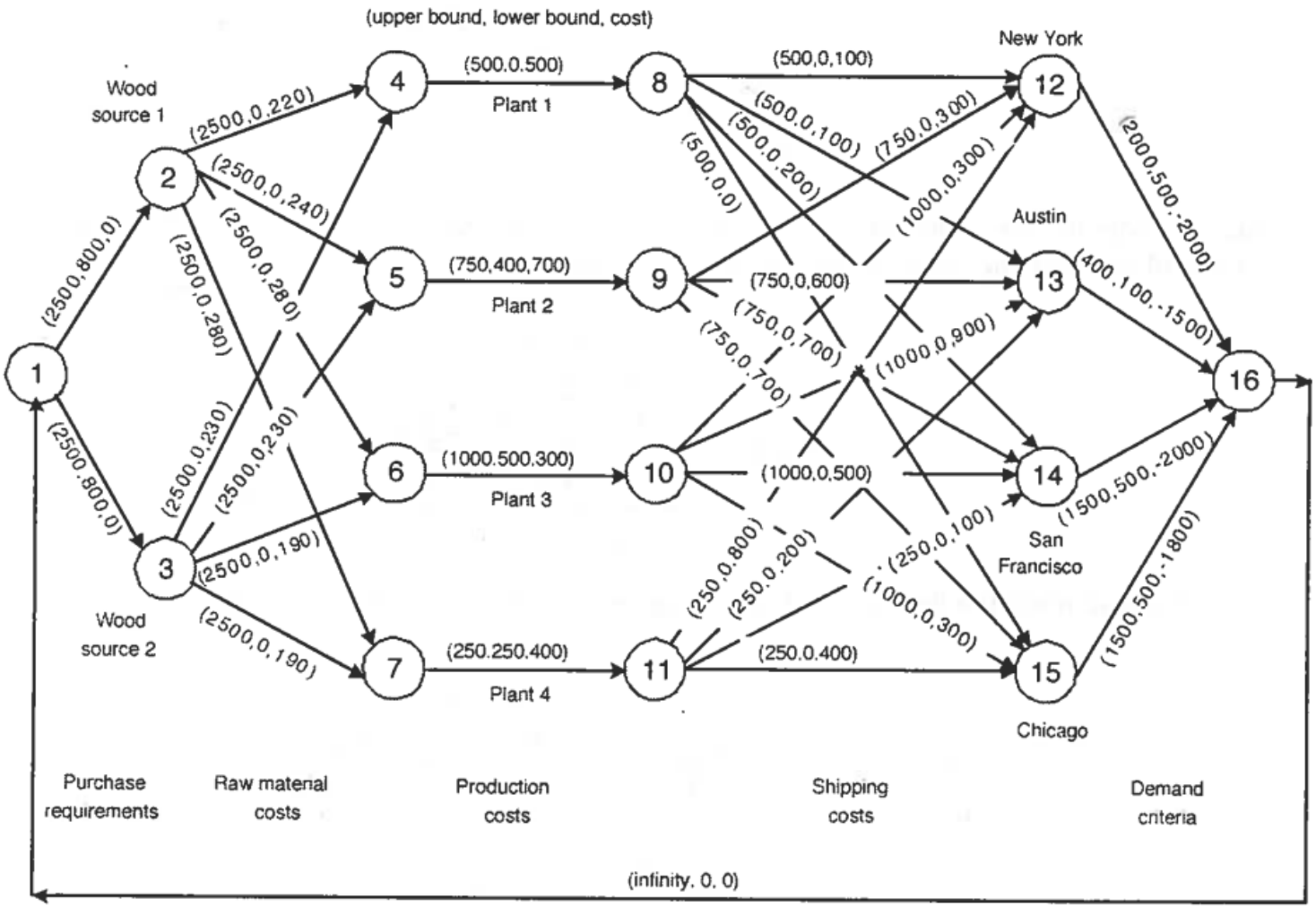
Minimum Cost Flow Model Example

Table 3.8. NETWORK LABELS

Arc	From Node	To Node	Upper Bound	Lower Bound	Cost
1	1	2	2500	800	0
2	1	3	2500	800	0
3	2	4	2500	0	220
4	2	5	2500	0	240
5	2	6	2500	0	280
6	2	7	2500	0	280
7	3	4	2500	0	230
8	3	5	2500	0	210
9	3	6	2500	0	190
10	3	7	2500	0	190
11	4	8	500	0	500
12	5	9	750	400	700
13	6	10	1000	500	300
14	7	11	250	250	400
15	8	12	500	0	100
16	8	13	500	0	100
17	8	14	500	0	200
18	8	15	500	0	0
19	9	12	750	0	300
20	9	13	750	0	600
21	9	14	750	0	700
22	9	15	750	0	300
23	10	12	1000	0	300
24	10	13	1000	0	100
25	10	14	1000	0	500
26	10	15	1000	0	300
27	11	12	250	0	800
28	11	13	250	0	200
29	11	14	250	0	100
30	11	15	250	0	400
31	12	16	2000	500	−2000
32	13	16	400	100	−1500
33	14	16	1500	500	−2000
34	15	16	1500	500	−1800
35	16	1	5400	1600	0

Number of nodes = 16
Number of arcs = 35

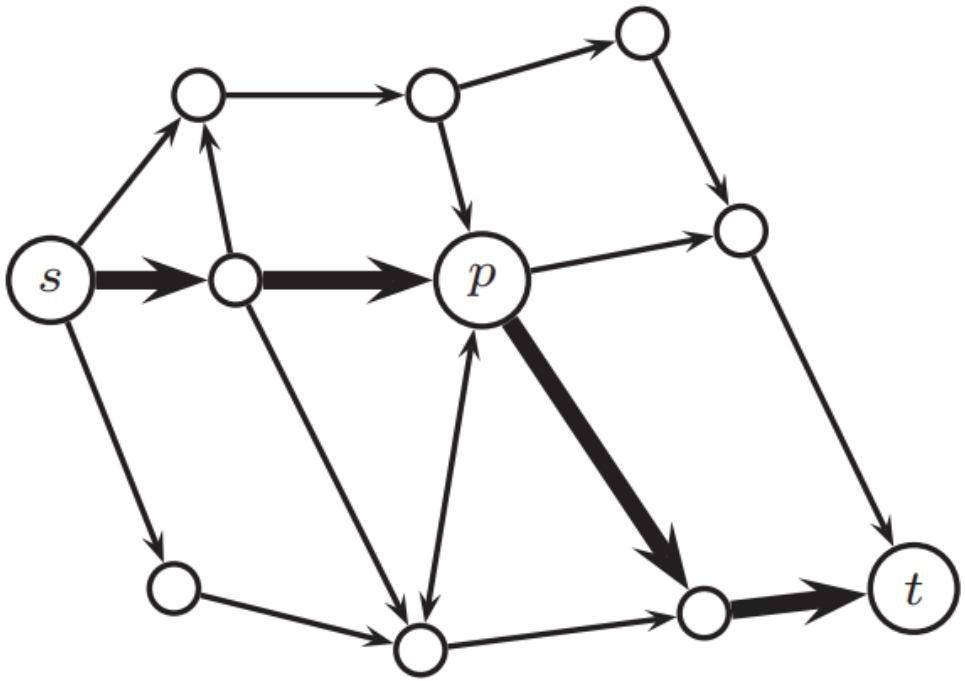
Minimum Cost Flow Model Example



Shortest Path Problem

- Directed graph $D = (V, A)$ with $s, t \in V$
- $b_s = 1$; $b_t = -1$
- $c_a \geq 0$ for $a \in A$

Need to find the shortest path from s to t (if $x_{ij} = 1$ in the solution, the arc belongs to the shortest path)



Dynamic Programming Principle

- Obs 5.1: if shortest path $s \to t$ passes through p then subpaths $s \to p$ and $p \to t$ are the shortest
- Obs 5.2: Let $d_k(v)$ be he length of shortest path from s to v containing at most k arcs then

$$d_k(v) = \min \left\{ d_{k-1}(v), \min_{i \in V^-(v)} \{ d_{k-1}(i) + c_{iv} \} \right\}$$

- Def: **Dynapmic Programming (DP) principle:** The optimal solution value for one problem is calculated recursively from the optimal values of simpler instances of the same problem.
- Def: The recursive property that pieces the optimal solutions is the **Principle of Optimality**

See Chapter 5 for more examples

Submodularity

Given a set function $f : 2^N \to \mathbb{R}$

- Def: *Submodular integer optimization problem:* $\max_{x \in \mathbb{Z}^n_+} \{cx \ : \ \sum_{j \in S} x_j \leq f(S) \text{ for } S \subseteq N\}$

Recall greedy heuristic:

- Order $c_1 \geq \ldots c_r > 0 \geq c_{r+1} \geq c_n$
- Set $x_i = f(\{1, \ldots i\}) - f(\{1, \ldots i-1\})$ if $i \leq r$ and $x_i = 0$ otherwise

What are the conditions on f under which the greedy heuristic works?

Submodularity

- Def 3.5: $f : 2^N \to \mathbb{R}$ is **submodular** if for all $A, B \subseteq N$

$$f(A) + f(B) \geq f(A \cap B) + f(A \cup B)$$

Equivalently, for all $A \subset B \subseteq N$ and $j \notin B$

$$f(A \cup \{j\}) - f(A) \geq f(B \cup \{j\}) - f(B)$$

Example: $f(S) = \phi(|S|)$ where ϕ is a concave function. (submodular function behaves more like convex)

- Def 3.5: $f : 2^N \to \mathbb{R}$ is **nondecreasing** if $f(A) \leq f(B)$ for all $A \subset B \subseteq N$
- Th 3.6: Submodular integer optimization with submodular, nondecreasing and integer-valued f is solvable with greedy heuristic.

Submodularity Example: Maximum Spanning Tree

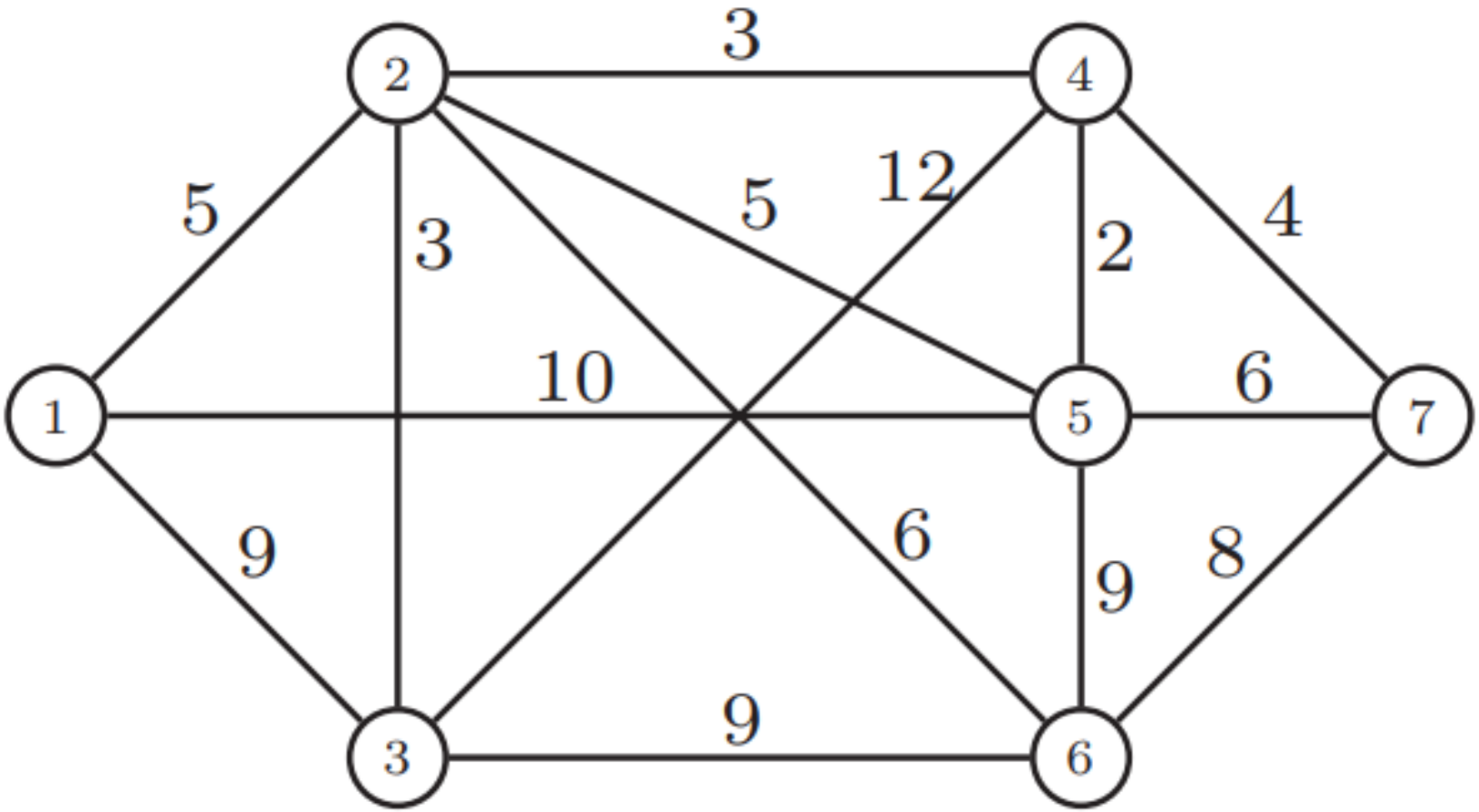
We are planning to set up a telecommunication network. The cost of a line between locations i and j is $-c_{ij}$. How to create a connected network optimally?

- Graph $G = (V, E)$
- Edge weights c_e for $e \in E$
- Find a maximum weight subgraph that is a tree

$$\max_{x \in \mathbb{Z}_+^{|E|}} \sum_{e \in E} c_e x_e$$
$$\text{s.t.} \quad \sum_{e \in E(S)} x_e \leq |S| - 1 \text{ for } S \subseteq V$$

where $E(S) = \{(i,j) \in E \ : \ i,j \in S\}$

Maximum Spanning Tree: solution example



Maximum Spanning Tree: solution example

	e_1	e_2	e_3	e_4	e_5	e_6	e_7
(i,j)	$(3,4)$	$(1,5)$	$(1,3)$	$(3,6)$	$(5,6)$	$(6,7)$	$(5,7)$
c_e	12	10	9	9	9	8	6
	+	+	+	+		+	

e_8	e_9	e_{10}	e_{11}	e_{12}	e_{13}	e_{14}
$(2,6)$	$(1,2)$	$(2,5)$	$(4,7)$	$(2,3)$	$(2,4)$	$(4,5)$
6	5	5	4	3	3	2
+						

.

References: Trees and Forests

- Wiki: Tree
- Graph theory (for java programmers but nice simple explanations)
- Lecture slides: Trees
- Trees and Connectivity
- Wolsey: Optimal Trees and OR

Complexity summary: easy problems

There are polynomial algorithms for the following optimization problems

- The Uncapacitated Lot-Sizing Problem (Chapter 5)
- The Assignment Problem (Chapter 4)
- The Maximum Weight Matching Problem (Chapter 4)
- The Shortest Path Problem (Chapter 5)
- The Max Flow Problem (Chapter 3)
- The Maximum Weight Tree Problem (Chapter 3)
- The TU Integer Programming Problem (Chapter 3)

Textbook: Section 6.1

Complexity summary: hard problems

No known efficient algorithms exist for the following optimization problems

- The 0–1 Knapsack Problem (Chapter 1)
- The Set Covering Problem (Chapter 1)
- The Traveling Salesman Problem (Chapter 1)
- The Uncapacitated Facility Location Problem (Chapter 1)
- The Integer Programming Problem (Chapter 1)
- The Steiner Tree Problem (Chapter 3)
- The Cutting Stock Problem (Chapter 5)

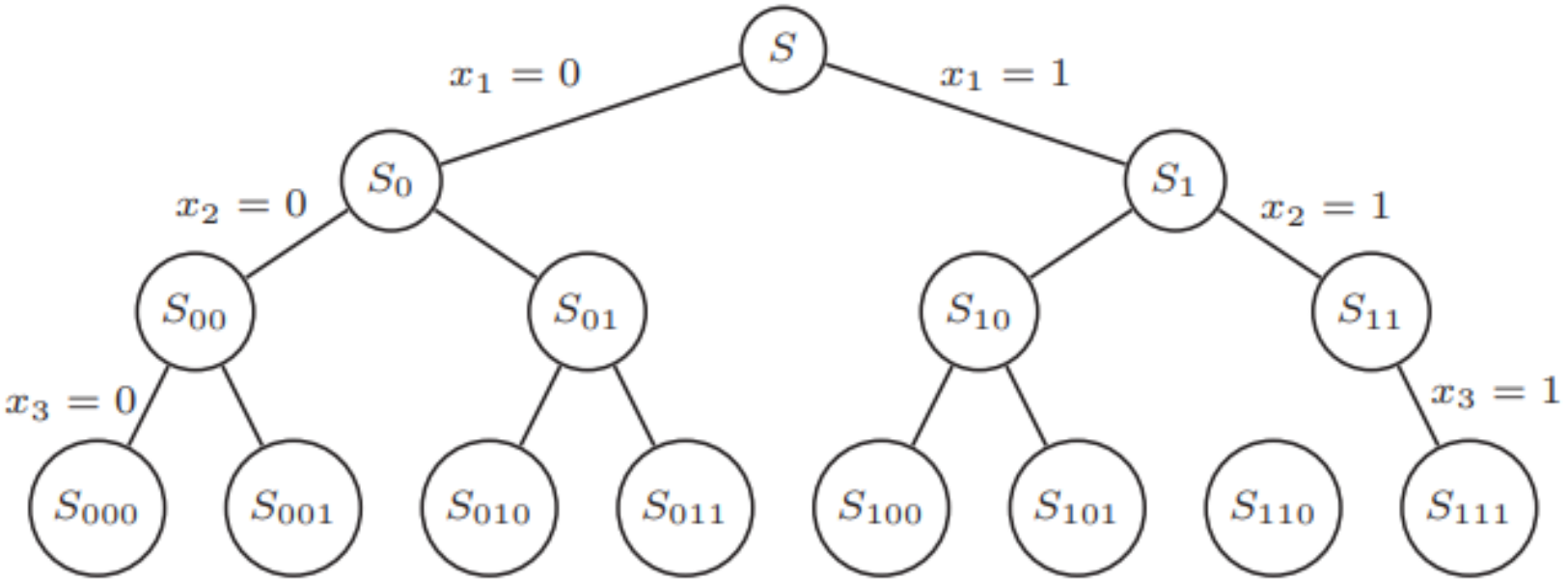
Textbook: Section 6.1

Divide and conquer

- Prop 7.1: Let $S = S_1 \cup \ldots X_K$ and $Z^k = \max\{cx \ : \ x \in S_k\}$ then

$$\max_k Z^k = Z = \max\{cx \ : \ x \in S\}$$

Example: enumeration tree. Suppose $S = P \cap \{0,1\}^3$. Divide it into $S_0 = \{x \in S \ : \ x_1 = 0\}$ and $S_1 = \{x \in S \ : \ x_1 = 1\}$, then divide S_0 into $S_{00} = \{x \in S_0 \ : \ x_2 = 0\}$ and $S_{01} = \{x \in S \ : \ x_1 = 0, x_2 = 1\}$...



Combinatorial Explosion

A top supercomputer Fugaku would take 10^{265} ages of the Universe to enumerate a COP if $n = 1000$

n	$\log n$	$n^{0.5}$	n^2	2^n	$n!$
10	3.32	3.16	10^2	1.02×10^3	3.6×10^6
100	6.64	10.00	10^4	1.27×10^{30}	9.33×10^{157}
1000	9.97	31.62	10^6	1.07×10^{301}	4.02×10^{2567}

Implicit enumeration

Prop 7.2: Let $S = S_1 \cup \ldots \cup S_K$ and $\underline{Z}^k \leq Z^k = \max_{x \in S_k} cx \leq \overline{Z}^k$ then

$$\underline{Z} = \max_k \underline{Z}^k \leq Z = \max_{x \in S} cx \leq \max_k \overline{Z}^k = \overline{Z}$$

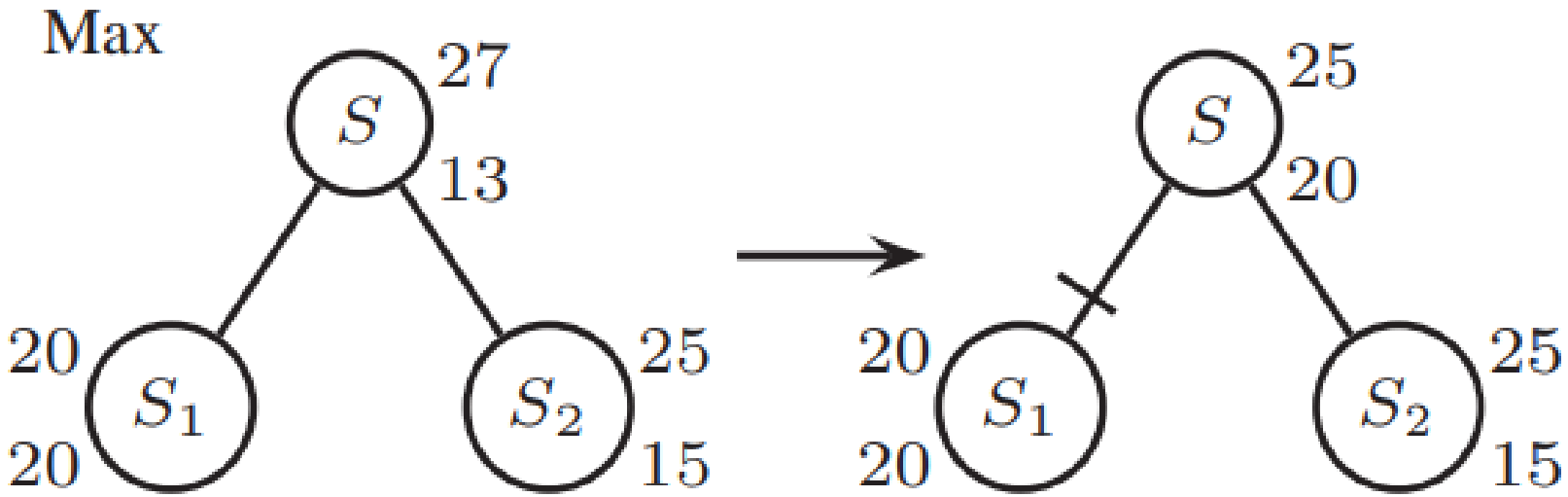
Idea: obtain upper and lower bounds for nodes instead of solving exactly

Pruning: do not branch a node S_k node

- Pruning by optimality: Z^k has been found
- Pruning by bound: $\overline{Z}^k \leq \underline{Z}$
- Pruning by infeasibility: $S_k = \emptyset$

Example: Pruning by optimality

- update
  - $\overline{Z} = \max_k \overline{Z}^k = \max\{20, 25\} = 25$
  - $\underline{Z} = \max_k \underline{Z}^k = \max\{20, 15\} = 20$
  - updete bounds on Z : $(13, 27) \rightarrow (20, 25)$
- $20 = \underline{Z}^1 \leq Z^1 \leq \overline{Z}^1 = 20$ so $Z^1 = 20$ and we prune Z^1 by optimality
- S_2 needs further examination



Example: Pruning by infeasibility

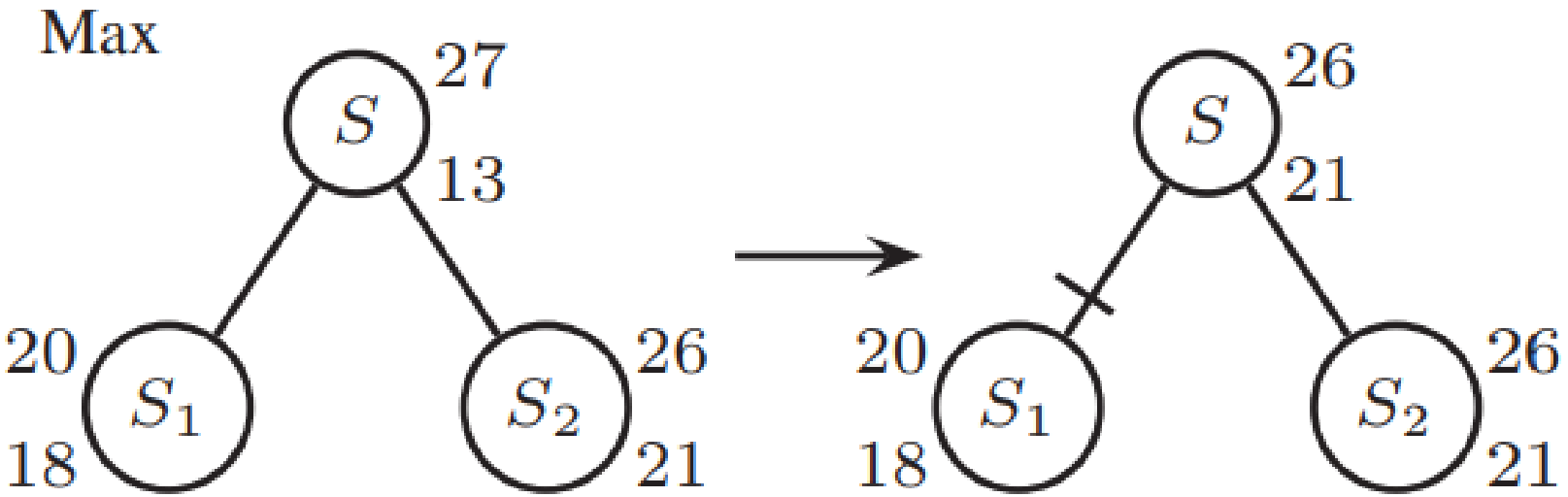
Actually, this is a special case of pruning by optimality as $\underline{Z}^k = \overline{Z}^k = Z^k = -\infty$

$$\begin{aligned} \min_{x_1, x_2 \in \{0,1\}} \quad & 2x_1 - 3x_2 \\ \text{s.t.} \quad & x_1 \geq 0.5 \end{aligned}$$

(S_0 is infeasible and needs not to be branched)

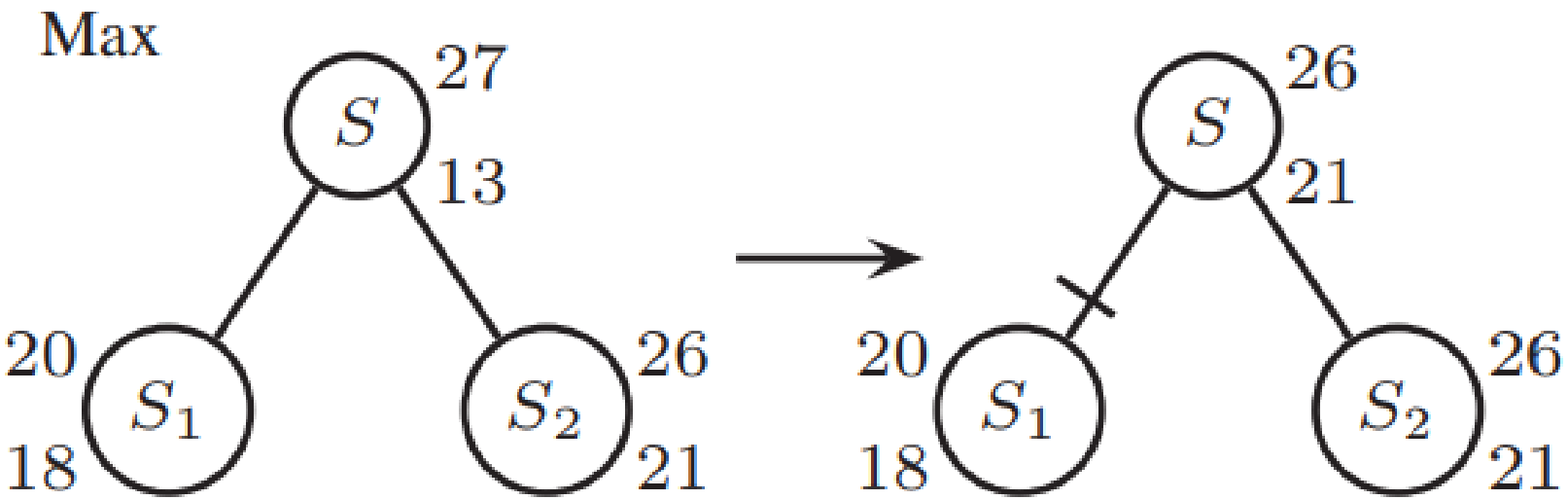
Example: Pruning by bound

- update
  - $\overline{Z} = \max_k \overline{Z}^k = \max\{20, 26\} = 26$
  - $\underline{Z} = \max_k \underline{Z}^k = \max\{18, 21\} = 21$
  - updete bounds on Z : $(13, 27) \rightarrow (21, 26)$
- Now $\underline{Z} = 21 > 20 = \overline{Z}^1$ and so the optimal solution cannot belong to S_1, thus we prune it by bound
- S_2 needs further examination



Example: No pruning possible

- update
  - $\overline{Z} = \max_k \overline{Z}^k = \max\{24, 37\} = 37$
  - $\underline{Z} = \max_k \underline{Z}^k = \max\{13, -\infty\} = 13$
  - updete bounds on Z : $(-\infty, 40) \rightarrow (13, 37)$
- S_2 needs further examination
- S_2 needs further examination



Research questions to implicit enumeration

- Should we put computation time into obtaining better bounds on the nodes or into branching of the nodes?
- How to make the separation $S = S_1 \cup \ldots \cup S_K$?
- In what order to examine subproblems?

Example: implicit enumeration + dual bounds

- We want to solve the following integer linear program:

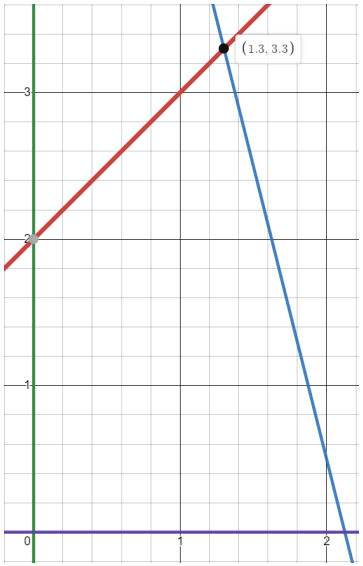
$$\begin{aligned} \min_{x} \quad & -5.5x_1 - 2.1x_2 \\ \text{s.t.} \quad & -x_1 + x_2 \leq 2 \\ & 8x_1 + 2x_2 \leq 17 \\ & x_1, x_2 \in \mathbb{Z}_+ \end{aligned}$$

- Exploit the idea of explicit enumeration
- Use LP relaxations to obtain dual bounds

Root node: Solve LP relaxation

root LP relaxation:

$$\begin{aligned} \min_{x \in \mathbb{R}^2_+} \quad & -5.5x_1 - 2.1x_2 \\ \text{s.t.} \quad & -x_1 + x_2 \leq 2 \\ & 8x_1 + 2x_2 \leq 17 \end{aligned}$$



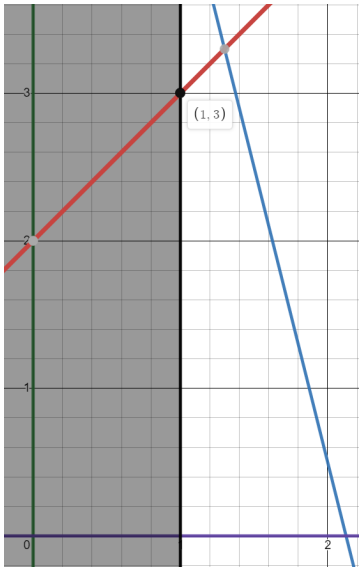
root

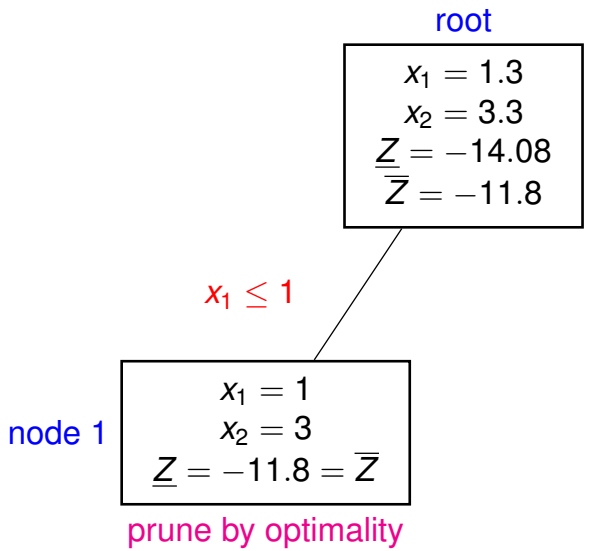
$x_1 = 1.3$
$x_2 = 3.3$
$\underline{Z} = -14.08$

Fork root into $x_1 \leq 1$ and $x_1 \geq 2$

node 1 LP:

$$\begin{aligned} \min_{x \in \mathbb{R}^2_+} \quad & -5.5x_1 - 2.1x_2 \\ \text{s.t.} \quad & -x_1 + x_2 \leq 2 \\ & 8x_1 + 2x_2 \leq 17 \\ & x_1 \leq 1 \end{aligned}$$



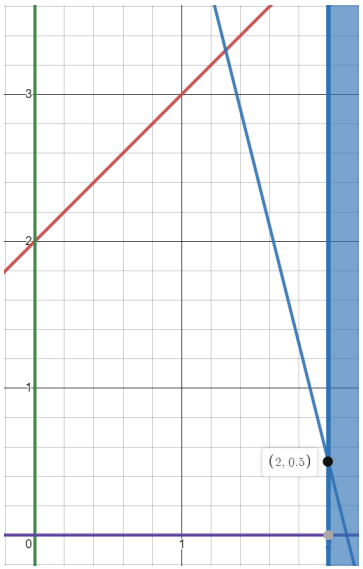


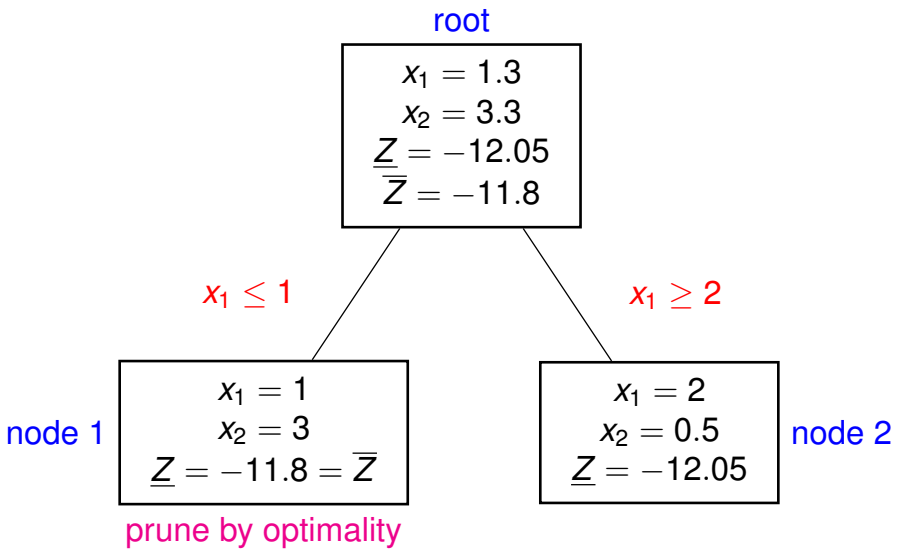
Fork root into $x_1 \leq 1$ and $x_1 \geq 2$

node 2 LP:

$$\min_{x \in \mathbb{R}^2_+} \; -5.5x_1 - 2.1x_2$$

$$\text{s.t.} \quad -x_1 + x_2 \leq 2$$
$$8x_1 + 2x_2 \leq 17$$
$$x_1 \geq 2$$





Fork node 2 into $x_2 \leq 0$ and $x_2 \geq 1$

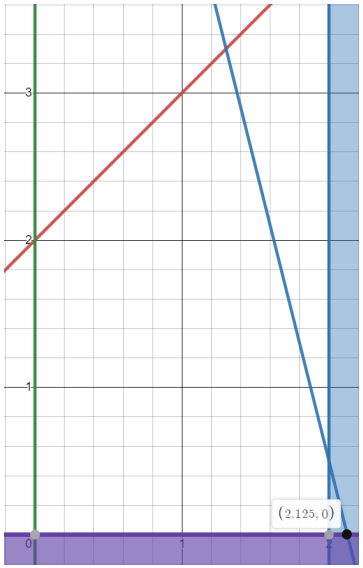
node 3:

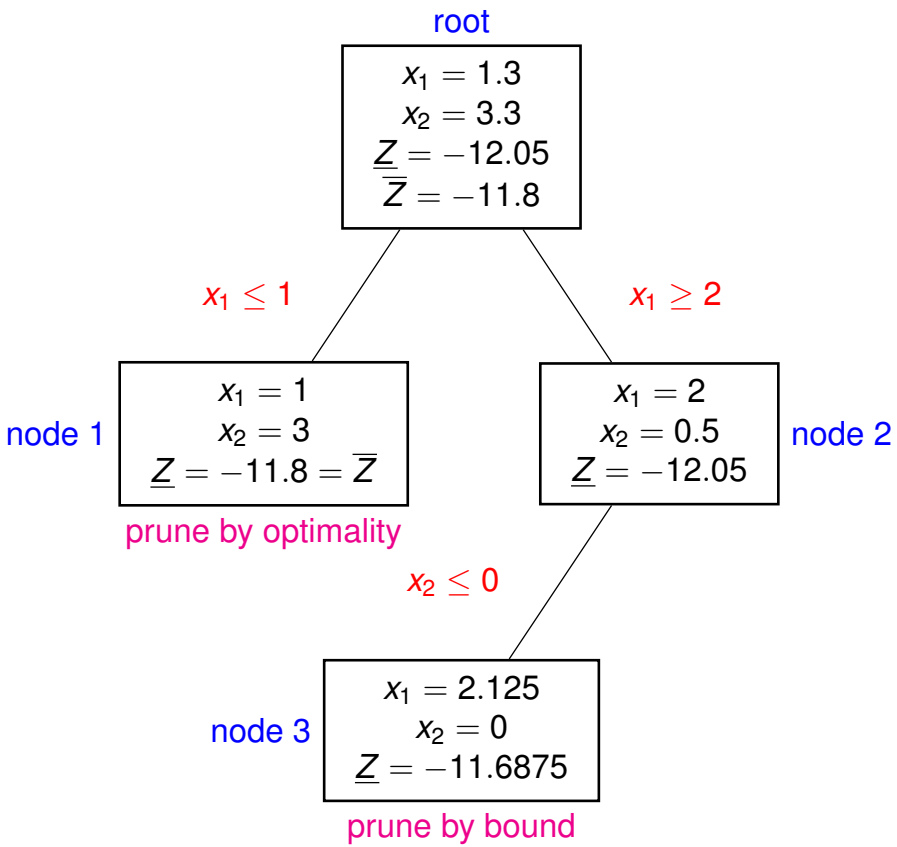
$$\min_{x \in \mathbb{R}^2_+} \quad -5.5x_1 - 2.1x_2$$

$$\text{s.t.} \quad -x_1 + x_2 \leq 2$$

$$8x_1 + 2x_2 \leq 17$$

$$x_1 \geq 2, x_2 \leq 0$$





Fork node 2 into $x_2 \leq 0$ and $x_2 \geq 1$

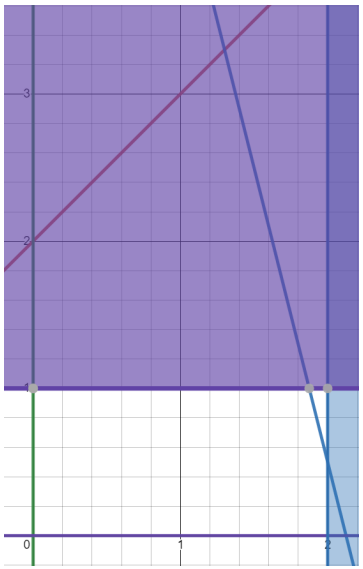
node 4:

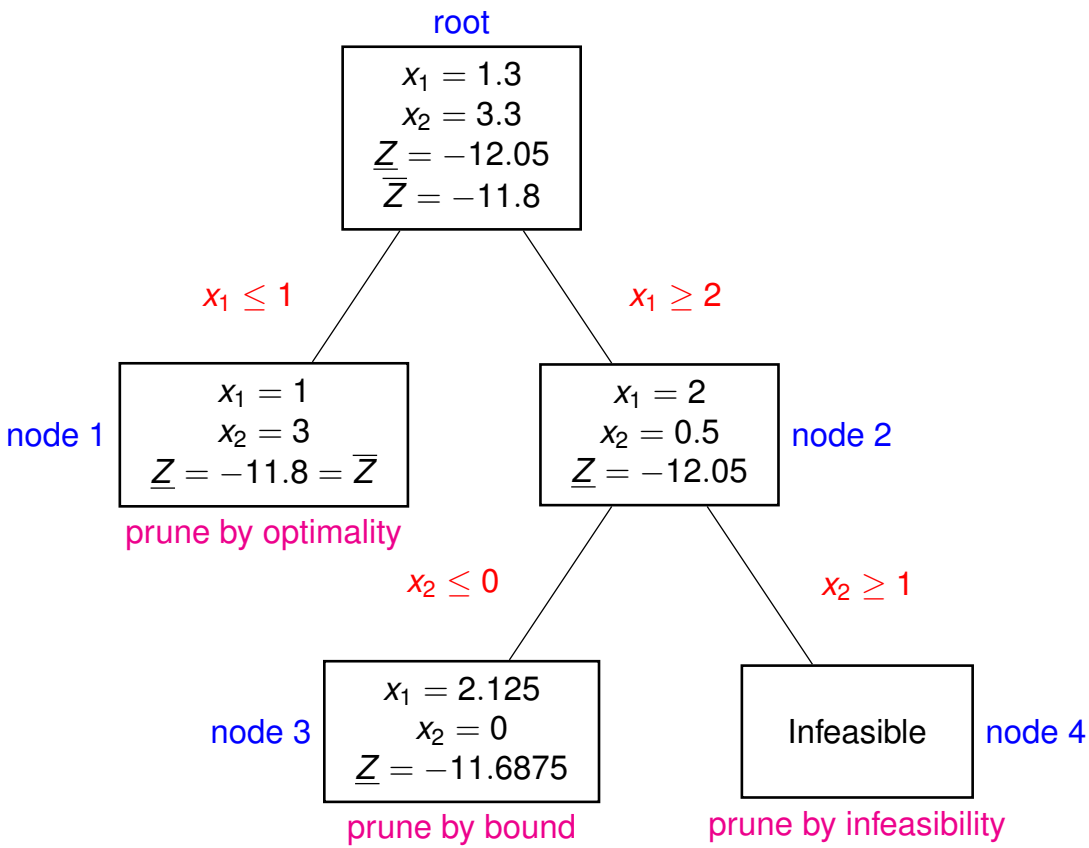
$$\min_{x \in \mathbb{R}^2_+} \; -5.5x_1 - 2.1x_2$$

$$\text{s.t.} \quad -x_1 + x_2 \leq 2$$

$$8x_1 + 2x_2 \leq 17$$

$$x_1 \geq 2, x_2 \geq 1$$





$$x_1^{IP} = 1, x_2^{IP} = 3, Z = -11.8$$

Branch-and-bound

- Branch-and-bound works by partitioning the search space, which is referred to as **branching**
- Pruning the enumeration by:
  - Optimality
  - Infeasibility
  - Bounding
- Specific strategy used here:
  - Variable branching: dichotomies of the type $x \leq \lfloor \bar{x} \rfloor$ xor $x \geq \lceil \bar{x} \rceil$
  - Linear programming bounding: using linear relaxations to compute dual bounds at each node
- Using the simplex method: resolving LPs is easier than starting from scratch. (All recent commercial codes include an interior point algorithm, as for many large linear programs, the latter algorithm is faster than the simplex method)

The branch-and-bound algorithm

(variable branching/LP bounding)

- Needs to maintain a list of LPs with additional constraints: $x_i \leq u_i$ or $x_i \geq l_i$
- Each LP corresponds to a node N_i of the enumeration tree, with value z_i
  (node N_0 corresponds to the linear relaxation, with value z^{LP})
- $\mathcal{L}$ is the list of nodes that must be solved, i.e. leaves of the enumeration tree that had not being pruned nor branched on
- $\underline{Z}$ is a lower bound on Z (optimal value) and $\bar{Z}$ is an upper bound on zZ
- $(x^{\text{best}}, y^{\text{best}})$ is the current best solution found, with value $\bar{Z}$

The branch-and-bound algorithm

(variable branching/LP bounding)

0. Initialize: $\mathcal{L} := \{N_0\}$, $\underline{Z} = -\infty$, $\bar{Z} = +\infty$, $(x^{\text{best}}, y^{\text{best}}) = 0$
1. Terminate?
   If $\mathcal{L} = \emptyset$, then $(x^{\text{best}}, y^{\text{best}})$ is optimal
   If $\bar{Z} - \underline{Z} \leq \epsilon$, then $(x^{\text{best}}, y^{\text{best}})$ is ϵ-suboptimal
2. Select a node N_i from $\mathcal{L}$ and delete it from $\mathcal{L}$
3. Bound:
   Solve N_i. If it is infeasible, go to Step 1. Else, let (x^i, y^i) be an optimal solution of N_i with value $\underline{Z}_i$.
   If $\underline{Z}_i < \underline{Z}$, update $\underline{Z}$
4. Prune:
   If $\underline{Z}_i \geq \bar{Z}$, go to Step 1.
   If (x^i, y^i) is feasible to MILP, set $\bar{Z} := \underline{Z}_i$, $(x^{\text{best}}, y^{\text{best}}) := (x^i, y^i)$ and go to Step 1.

The branch-and-bound algorithm

(variable branching/LP bounding)

5. Branch:
   From N_i construct $k \geq 2$ LPs $N_{i_1}, \ldots, N_{i_k}$ with smaller feasible regions, whose union does not contain (x^i, y^i), but contains all the solutions of N_i with $x \in \mathbb{Z}$.
   Add $N_{i_1}, \ldots, N_{i_k}$ to $\mathcal{L}$ and go to Step 1.

Several issues require especial attention when using branch-and-bound algorithms:

- Formulations (the better the formulation, the less branching)
- Heuristics (to find good solutions without diving on the enumeration tree)
- Node selection
- Branching strategy

Node selection

- Two conflicting criteria:
  - Pruning the tree requires good solutions providing tight upper bounds. One should descend as quick as possible on the tree in order to find feasible solutions $\rightarrow$ **depth-first search**
  - In order to prove optimality we need good bounds. One should choose the node with the best (smallest lower) bound $\rightarrow$ **best-first search**
- Practical branch-and-bound implementations use a combination of the two
  1. Depth-first search provides an initial feasible solution
  2. Mix of depth-first (finding better solutions) and best-first (providing better bounds)

Branching

- Variable branching:
  - Most-fractional branching
    Select variable x_i with largest $\min\{\lceil x_i^* \rceil - x_i^*, x_i^* - \lfloor x_i^* \rfloor\}$
  - Strong branching
    1. Pre-branch up and down on all fractional variables. For each fractional variable i we obtain down lower bound z_i^D and up lower bound z_i^U.
    2. Select variable with largest effect on lower bound to branch-on, i.e. $i^{\text{branch}} = \arg\max_i \min\{z_i^D, z_i^U\}$
  - Pseudocost branching
    Use strong branching idea, but replace z_i^D and z_i^U with estimates from the tree
- Generalized Upper Bound (GUB) or Special Ordered Sets (SOS) — Section 7.5

LP pre-solving example

$$
\begin{aligned}
\max_x \; & 2x_1 + x_2 - x_3 \\
\text{s.t.} \; & 5x_1 - 2x_2 + 8x_3 \leq 15 \\
& 8x_1 + 3x_2 - x_3 \geq 9 \\
& x_1 + x_2 + x_3 \leq 6 \\
& 0 \leq x_1 \leq 3 \\
& 0 \leq x_2 \leq 1 \\
& 1 \leq x_3
\end{aligned}
$$

See LP (Prop. 7.5) and IP (Prop. 7.6) pre-solving techniques

LP pre-solving example

- $5x_1 \leq 15 + 2x_2 - 8x_3 \leq 15 + 2 \times 1 - 8 \times 1 = 9 \Rightarrow x_1 \leq \frac{9}{5}$
- $8x_3 \leq 15 + 2x_2 - 5x_1 \leq 15 + 2 \times 1 - 5 \times 0 = 17 \Rightarrow x_3 \leq \frac{17}{8}$
- $8x_1 \geq 9 - 3x_2 + x_3 \geq 9 - 3 + 1 = 7 \Rightarrow x_1 \geq \frac{7}{8}$
- $8x_3 \leq 15 + 2x_2 - 5x_1 \leq 15 + 2 - 5 \times \frac{7}{8} = \frac{101}{8} \Rightarrow x_3 \leq \frac{101}{64}$
- $x_1 + x_2 + x_3 \leq \frac{9}{5} + 1 + \frac{101}{64} < 6 \Rightarrow$ redundant
- increasing $x_2 \Rightarrow$ relaxes constraints (other than the bound), improves the objective. Thus, $x_2 = 1$ (its upper bound)
- decreasing $x_3 \Rightarrow$ relaxes constraints (other than the bound), improves the objective. Thus, $x_3 = 1$ (its lower bound)

$$\max\{2x_1 \ : \ \frac{7}{8} \leq x_1 \leq \frac{9}{5}\}$$

The cutting-plane method

- Def 8.1: $\pi x \leq \pi_0$ is a **valid inequality** for X if $\pi x \leq \pi_0$ for all $x \in X$
- Def 8.2: $\pi x \leq \pi_0$ is a **cut** of $x^0 \in \text{conv}(X)$ from X it is valid for X but $\pi x^0 > \pi_0$
- Consider the mixed-integer program

$$\max_{x,y} cx + hy$$
$$\text{s.t. } (x, y) \in S = \{(u, v) \in \mathbb{Z}^n_+ \times \mathbb{R}^p_+ \mid Au + Gv \leq b\}$$

- Let P be the linear relaxation of S, with optimal value Z^{LP} and optimal (basic) solution $(x^{LP}, y^{LP}) \notin S$ for the problem at hand
- Alternative to branching: Find a cut for (x^{LP}, y^{LP}).
- Cutting planes are guaranteed to exists (if (x^{LP}, y^{LP}) is a basic solution)

The cutting-plane method

0. Let P be the linear relaxation of S
1. Let

$$(\tilde{x}, \tilde{y}) = \arg\max_{x,y} cx + hy$$
$$\text{s.t. } (x, y) \in P$$

2. If $(\tilde{x}, \tilde{y}) \in S$, terminate. $(\tilde{x}, \tilde{y})$ is optimal.
3. Find a cutting plane $\alpha x + \gamma y \leq \beta$ separating $(\tilde{x}, \tilde{y})$ from S
4. Set $P := P \cap \{(u, v) \in \mathbb{R}^n \times \mathbb{R}^p \mid \alpha u + \gamma v \leq \beta\}$ and return to 1.

Branch-and-cut

- Putting together branch-and-bound and cutting-plane methods
- Cplex, Gurobi, Xpress, SCIP, GLPK, CBC, and other mixed-integer programming system, all use some form of branch-and-cut
- Extra step, after pruning (Step 4) and before branching (Step 5):
  Add cuts to strengthen the formulation of N_i? or branch?
  In the first case, add cuts and return to bounding (Step 3).
  Otherwise, proceed to branching (Step 5).
- Classes of cuts:
  - Gomory cuts
  - Mixed Integer Rounding
  - Split cuts
  - Dantzig-Wolf cuts (Chapter 11)
  - Benders' cuts (Chapter 12)
  - Problem-specific cutting planes (Section 14.4)

Chavatal-Gomory procedure

Th: Let $P = \{x \in \mathbb{R}^n_+ \ : \ Ax \leq b\}$ where $A = [a_1, \dots a_n] \in \mathbb{R}^{m \times n}$, consider $X = P \cap \mathbb{Z}^n$ and $u \in \mathbb{R}^m_+$. The inequalities are valid for X:

- $\sum_{j=1}^n ua_j x_j \leq ub$
- $\sum_{j=1}^n \lfloor ua_j \rfloor x_j \leq ub$
- $\sum_{j=1}^n \lfloor ua_j \rfloor x_j \leq \lfloor ub \rfloor$

Th 8.1: Every valid inequality for X can be obtained through a sequence of Chavatal-Gomory inequalities

Example: Chavatal-Gomory procedure

$$\begin{aligned} 7x_1 - 2x_2 &\leq 14 \\ x_2 &\leq 3 \\ 2x_1 - 2x_2 &\leq 3 \\ x &\geq 0 \end{aligned}$$

$u = (\frac{2}{7}, \frac{37}{63}, 0)$

- $2x_1 + \frac{1}{63}x_2 \leq \frac{121}{21}$
- $2x_1 + 0x_2 \leq \frac{121}{21}$
- $2x_1 \leq \lfloor \frac{121}{21} \rfloor = 5$
- $x_1 \leq \lfloor \frac{5}{2} \rfloor = 2$

Machine Learning for IP

One might hope to design an ML system that, given a dataset of instance-solution pairs, outputs an effective heuristic to tackle the problem instances from the underlying distribution. Aspects to learn include:

- branching rules that attempt to approximat strong branching, but run much faster
- rules to select a branching variable
- rules to decide whether a heuristic should be called
- rules to select which Gomory cuts should be added
- a rule to decide whether a decomposition algorithm should be used on a given problem/instance

IP for Machine Learning

The core of machine learning is optimization but IP is not a favorable paradigm of learning these days (continuous black-box optimization is).
Nevertheless, IP-based approaches are used in specific contexts:

- Column generation algorithm may generate a short set of logical rules to classify a set of $0-1$ vectors (good for interpretability)
- MIP can be used to construct decision trees with linear decision rules for clasterization problem
- Search for adversarial examples for an ML system can be formulated as MIP