MS-C2105 - Introduction to Optimization
Lecture 7

Fabricio Oliveira (with modifications by Harri Hakula)

Systems Analysis Laboratory
Department of Mathematics and Systems Analysis

Aalto University
School of Science

March 18, 2022

Outline of this lecture

Modelling with integer variables

Fixed cost

Disjunctions and implications

Solving general IPs

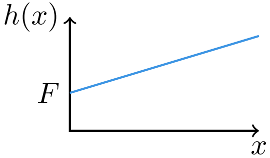
Branch-and-bound method

Reading: Taha: Chapter 9 (9.2); Winston: 9 (from 9.3)

Modelling with integer variables: fixed costs

In many problems, costs are composed of a fixed charge F plus a proportional charge p. The cost function h with fixed costs can be

$$h(x) = \begin{cases} F + px, & \text{if } 0 < x \leq C \\ 0, & \text{if } x = 0 \end{cases}$$



If we want to minimise $h(x)$, we can define $y \in \{0, 1\}$ such that $y = 1$, if $x > 0$, and $y = 0$, otherwise. This can be modelled as:

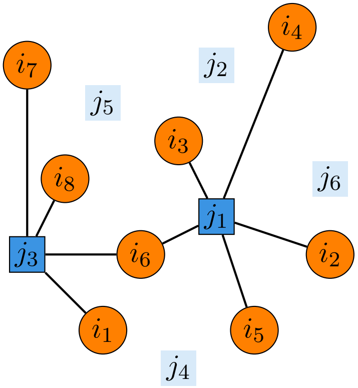
$$\begin{aligned} \min_{x,y} \quad & Fy + px \\ \text{s.t.:} \quad & x \leq Cy \\ & x \geq 0 \\ & y \in \{0, 1\} \end{aligned}$$

Uncapacitated facility location

Problem statement:

- $M = \{1, \dots, m\}$ clients must be served by a subset of $N = \{1, \dots, n\}$ facilities;
- opening a facility at location $j \in N$ has a fixed cost F_j;
- serving a client $i \in M$ by a facility $j \in N$ costs C_{ij}.

Objective: decide where to locate facilities and how to serve clients minimising the total (opening + service costs) cost.



Uncapacitated facility location

Define the following variables:

- x_{ij} be the fraction of the demand $i \in M$ being served by facility $j \in N$;
- $y_j = 1$, if a facility is opened at $j \in N$, and 0, otherwise.

The UFL can be formulated as:

$$
\begin{aligned}
\text{(UFL)} : \min_{x,y} \ & \sum_{j \in N} F_j y_j + \sum_{i \in M} \sum_{j \in N} C_{ij} x_{ij} \\
\text{s.t.:} \ & \sum_{j \in N} x_{ij} = 1, \forall i \in M \\
& \sum_{i \in M} x_{ij} \leq m y_j, \forall j \in N \\
& x_{ij} \geq 0, \forall i \in M, \forall j \in N \\
& y_j \in \{0, 1\}, \forall j \in N.
\end{aligned}
$$

Modelling disjunctions

Suppose that $x \in \mathbb{R}^n : 0 \leq x \leq u$ and we wish to impose:

$$\sum_{j=1}^{n} a_j^1 x_j \leq b^1 \vee \sum_{j=1}^{n} a_j^2 x_j \leq b^2$$

This disjunctive conditions occur often, whether condition 1 or 2 can happen, but not simultaneously. How can we model this?

Let $y_i \in \{0,1\}, i \in \{1,2\}$. We assume to have an upper bound $M_i \geq \left\{a^i x - b^i : 0 \leq x \leq u\right\}, i \in \{1,2\}$. Then we have:

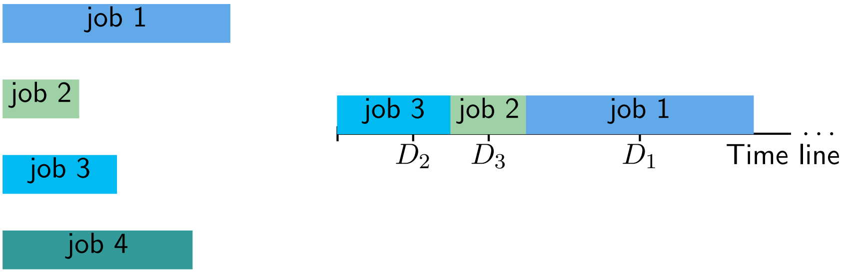
$$\begin{aligned}
&\sum_{j=1}^{n} a_j^i x_j - b^i \leq M_i(1-y_i), i \in \{1,2\} \\
&y_1 + y_2 \leq 1 \\
&y_i \in \{0,1\}, i \in \{1,2\}, 0 \leq x \leq u.
\end{aligned}$$

Scheduling

Problem statement: define the order jobs must be performed.

- jobs $j \in N = \{1, \dots, n\}$ are performed sequentially;
- each job j has due date D_j;
- performing job j takes P_j units of time (e.g., hours or days)
- a penalty C_j is paid for delay per unit of time

Objective: schedule jobs such that lateness penalty is minimised.



Scheduling

Consider the variables:

- x_j is the time job j starts.
- $s_j = s_j^+ - s_j^-$ represent the deviation from deadline D_j, both earliness (s_j^+) or lateness (s_j^-)

An important feature to consider is sequencing.

- if job i is schedule before job j, then $x_j \geq x_i + P_i$.
- otherwise, $x_i \geq x_j + P_j$

This either-or condition can be modelled as a disjunction. Let $y_{ij} \in \{0, 1\}$ indicate whether job i is scheduled before job j. Then

$$
\begin{aligned}
My_{ij} + (x_i - x_j) &\geq P_j \\
M(1 - y_{ij}) + (x_j - x_i) &\geq P_i
\end{aligned}
$$

Scheduling

The scheduling problem can be modelled as:

$$
\begin{aligned}
\text{min. } z = & \sum_{j=1}^{n} C_j s_j^- \\
\text{s.t.: } & M y_{ij} + (x_i - x_j) \geq P_j, \forall i, j \in N, i < j \\
& M(1 - y_{ij}) + (x_j - x_i) \geq P_i, \forall i, j \in N, i < j \\
& x_j + P_j + (s_j^+ - s_j^-) = D_j \\
& x_j, s_j^+, s_j^- \geq 0, j \in N \\
& y_{ij} \in \{0, 1\}, \forall i, j \in N, i < j.
\end{aligned}
$$

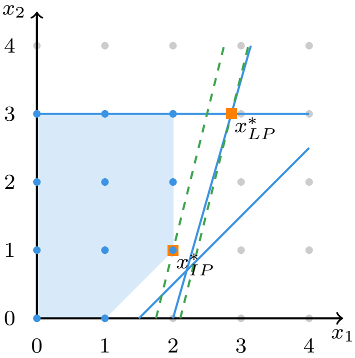
Remarks: several variants of objective function can be considered

- minimise plan deviation: min. $z = \sum_{j=1}^{n} s_j^+ + s_j^-$
- maximise earliness : max. $z = \sum_{j=1}^{n} s_j^+$

Solving IP problems

We explore the most popular exact method for solving IPs, which is based on two key concepts: linear relaxations and convex hulls.

A feasible region $S = \{Ax \leq b : x \in \mathbb{Z}_+\}$ is illustrated below.



- We use LP relaxations, which is the IP with integrality constraints removed.
- If the convex hull is available, the IP can be solved as an LP.

Solving IP problems

Branch-and-bound (B&B) is a divide-and-conquer strategy for solving (mixed-)integer programming problems such as

$$(P) : z_{IP} = \max_x . \left\{ c^\top x : x \in S \right\}.$$

The divide-and-conquer paradigm is based on the following idea:

1. Break P into subproblems (that might be easier to solve);
2. Combine all the subproblem solutions to form a solution to P.

The working principle is summarised by this proposition:

Proposition 1

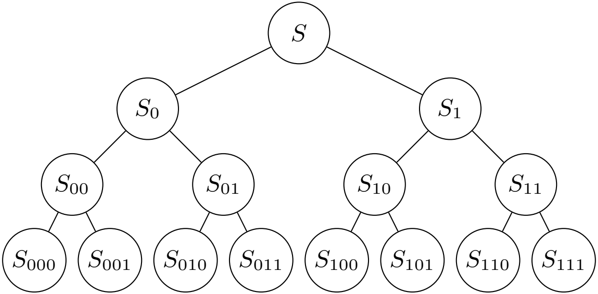
Let $K = \{1, \dots, |K|\}$ and $\bigcup_{k \in K} S_k = S$ be a decomposition of S. Let $z^k = \max._x \{cx : x \in S_k\}, \forall k \in K$. Then $z_{IP} = \max_{k \in K} \left\{z^k\right\}$

.

Solving IP problems

An important representation to control the generation of subproblems is a enumerative tree.

Example: Enumerative tree for $S \subseteq \{0,1\}^3$ (binary branching).



$S = S_0 \cup S_1 = \{x \in S : x_1 = 0\} \cup \{x \in S : x_1 = 1\}$

$S_i = S_{i0} \cup S_{i1} = \{x \in S : x_1 = i, x_2 = 0\} \cup \{x \in S : x_1 = i, x_2 = 1\}$

$S_{ij} = S_{ij0} \cup S_{ij1} = \{x \in S : x_1 = i, x_2 = j, x_3 = 0\} \cup \{x \in S : x_1 = i, x_2 = j, x_3 = 1\}$

The combinatorial explosion

Enumerative trees are only useful to organise the process. Fully enumerating all solutions is often hopeless.

1. **Assignment problem:** we have $n!$ permutations of $\{1, \ldots, n\}$.
2. **Knapsack and set covering problem:** maximum number of feasible subsets is 2^n.
3. **Travelling salesman problem:** starting from city 1, we have to check $(n-1)!$ permutations of $\{2, \ldots, n\}$.

n	2^n	$n!$
10	1.02×10^3	3.60×10^6
100	1.27×10^{30}	9.33×10^{157}
1000	1.07×10^{301}	4.02×10^{2567}

Table: Total number of iterations given input of size n

You can check how big these numbers are here.

The B&B method

General B&B methods rely on successively solving LP relaxations that are further constrained to generate subproblems.

- Subproblems are further constrained (branching) until becoming infeasible or returning a candidate integer solution.
- For maximisation, LP relaxations provide upper bounds ($\overline{z}$) while feasible (integer) solutions provide lower bounds ($\underline{z}$).

Branching: at a given subproblem S_k, suppose we have an optimal solution with a fractional component $\overline{x}_j$.

We can then branch S_k into the following subproblems:

$$S_{k1} = S_k \cap \{x : x_j \leq \lfloor \overline{x}_j \rfloor\}$$
$$S_{k2} = S_k \cap \{x : x_j \geq \lceil \overline{x}_j \rceil\}$$

The B&B method

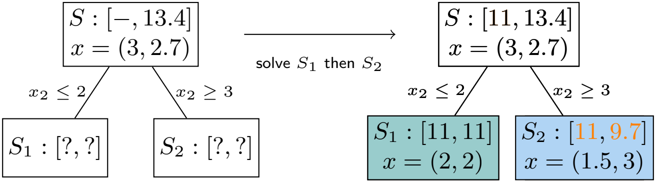
The efficiency of B&B is tied up with the bounding: avoiding fully investigate a branch to its leaves if bound information is available.

Proposition 2

Let $S = \bigcup_{k \in K} S_k$ *be a decomposition of* S *into smaller sets. Let* $z^k = \text{max.}_x \left\{c^\top x : x \in S_k\right\}$ *for* $k \in K$*, and let* $\overline{z}^k$ *(*$\underline{z}^k$*) be an upper (lower) bound on* z^k*. Then* $\overline{z} = \max_{k \in K} \overline{z}^k$ *and* $\underline{z} = \max_{k \in K} \underline{z}^k$.

Using the knowledge of global lower and local upper bounds, we can halt the search through S_k, i.e., prune S_k preemptively.

Example: Branching represented by edges and bounds by $[\underline{z}, \overline{z}]$.



Putting together a B&B method for IPs

Pruning (i.e., bounding) using information from the LP relaxation is possible in three distinct cases:

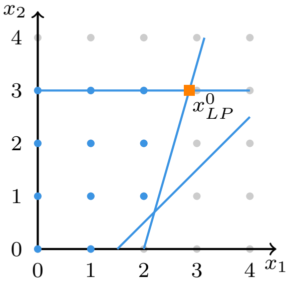
- **Pruning by optimality:** $z^k = \text{max.}_x \left\{c^\top x : x \in S_k\right\}$ is solved to optimality. If the solution of the LP relaxation is integer, we prune by optimality;
- **Pruning by infeasibility:** $S_k = \emptyset$. If the relaxation is infeasible, we prune by infeasibility.
- **Pruning by bound:** if $\overline{z}^k < \underline{z}$ (max. problem). If the solution of the relaxation provides a upper bound smaller than a known lower bound, we prune by bound.

Remark: pruning by bound requires a global lower bound. Thus, the sequence in which S_k are solved is crucial for performance.

B&B method for IPs: example

Consider the problem:

$$\max_x. \ z = 4x_1 - x_2$$
$$7x_1 - 2x_2 \leq 14$$
$$x_2 \leq 3$$
$$2x_1 - 2x_2 \leq 3$$
$$x_1, x_2 \in \mathbb{Z}_+$$

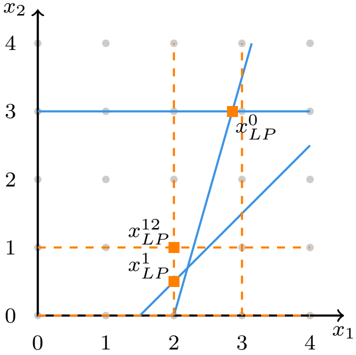


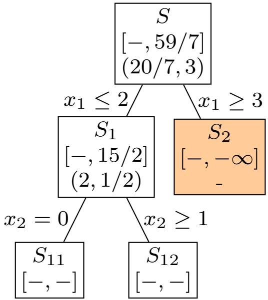
We start by solving the LP relaxation (bounding). At this point our tree is initialised as

$$\boxed{\begin{array}{c} S \\ [-, 59/7] \\ (20/7, 3) \end{array}}$$

B&B method for IPs: example

We choose to solve S_2. This leads to

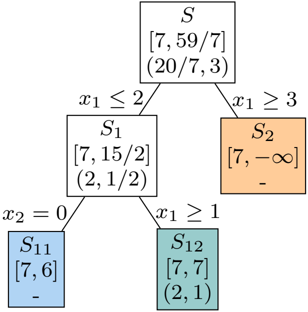




We prune S_2 by infeasibility and select another subproblem. We *arbitrarily* choose S_{12} which has an integer solution. Thus, we can prune S_{12} by optimality. Now $\mathcal{L} = \{S_{11}\}$.

B&B method for IPs: example

As we found an integer solution, we can update the global (primal) lower bound, i.e., $\underline{z} = \max\{-\infty, 7\} = 7$.



Finally, S_{11} can be pruned by bound. As $\mathcal{L} = \emptyset$, the algorithm is finished with $x^* = (2, 1)$, $z^* = 7$.

B&B method for IPs: example

Algorithm 1 shows the pseudocode for the LP-relaxation based B&B for a maximisation problem S with formulation P.

Algorithm LP-relaxation based B&B

1: **initialise.** $\mathcal{L} \leftarrow \{S\}$, $\underline{z} \leftarrow -\infty$, $\overline{x} \leftarrow -\infty$

2: **while** $\mathcal{L} \neq \emptyset$ **do**

3: select problem S_i from $\mathcal{L}$. $\mathcal{L} \leftarrow \mathcal{L} \setminus \{S_i\}$.

4: solve LP relaxation of S_i over P_i, obtaining z^i_{LP} and x^i_{LP}. $\overline{z}^i \leftarrow z^i_{LP}$.

5: **if** $S_i = \emptyset$ **then** return to step 2.

6: **else if** $\overline{z}^i \leq \underline{z}$ **then** return to step 2.

7: **else if** $x^i_{LP} \in \mathbb{Z}^n$ **then** $\underline{z} \leftarrow \max\left\{\underline{z}, \overline{z}^i\right\}$, $\overline{x} \leftarrow x^i_{LP}$; and return to step 2

8: **end if**

9: select a fractional component x_j and create subproblems S_{i1} and S_{i2} with formulations P_{i1} and P_{i2}, respectively, such that

$$P_{i1} = P_i \cup \{x_j \leq \lfloor \overline{x_j} \rfloor\} \text{ and } P_{i2} = P_i \cup \{x_j \leq \lceil \overline{x_j} \rceil\}.$$

10: $\mathcal{L} \leftarrow \mathcal{L} \cup \{S_{i1}, S_{i2}\}$.

11: **end while**

12: **return** $(\overline{x}, \underline{z})$.
