B6.3 Integer Programming

Prof. Raphael Hauser
Oxford Mathematical Institute

October 17, 2021

Table of Contents

1 Lecture 1: Modelling

1.1 What is integer programming?

Consider the following decision problem,

$$\begin{aligned}
&\max_{x_1,x_2,x_3\in\mathbb{R}} 3.4\,x_1 - 2\,x_2 + \sqrt{5}\,x_3\\
&\text{subject to } -7x_1 + x_3 \le 5,\\
&\qquad 2x_2 + x_3 = 6,\\
&\qquad x_1 \ge 0\\
&\qquad x_1, x_2, x_3 \in \mathbb{Z}.
\end{aligned}$$

It consists of the following parts.

- Decision variables: the variables x_1, x_2, x_3 whose values we are allowed to choose.
- An objective function $f(x_1, x_2, x_3) = 3.4x_1 - 2x_2 + \sqrt{5}x_3$ that we want to optimise (minimise or maximise).
- *Linear* constraints of the form $p(x) \le c$ or $p(x) = c$, where p is a polynomial of degree 1 in the decision variables and c a constant.
- *Integrality constraints* $x_i \in \mathbb{Z}$ for at least some of the decision variables.

Mathematical problems of this form are called *integer programming* (IP) problems if all decision variables are integrality constrained, or *mixed integer programming* problems if only some of the variables are are integrality constrained.

More generally, an IP is of the form

$$\begin{aligned}
&\max_{x\in\mathbb{R}^n} c^{\mathrm{T}}x\\
&\text{s.t. } Ax \le b, \quad \text{(componentwise)}\\
&\qquad x \ge 0, \quad \text{(componentwise)}\\
&\qquad x \in \mathbb{Z}^n,
\end{aligned}$$

where A is a $m \times n$ matrix for some $m, n \in \mathbb{N}$, and where $c \in \mathbb{R}^n$ and $b \in \mathbb{R}^m$ are vectors.

A problem such as

$$\begin{aligned}
&\max_{x\in\mathbb{R}^n} c^{\mathrm{T}}x\\
&\text{subject to } Ax \le b,\ x \ge 0,\ x \in \mathbb{Z}^n
\end{aligned}$$

refers to the maximum objective value achievable among the *feasible points* x, that is, by decision vectors that satisfy the constraints.

Often we are interested in the *maximising values* x^* of the decision variables themselves. In this case we write

$$\begin{aligned} x^* = \arg \max_{x\in\mathbb{R}^n} &\ c^{\mathrm{T}}x \\ \text{s.t. } &Ax \le b,\ x \ge 0,\ x \in \mathbb{Z}^n. \end{aligned}$$

This also makes it easy to solve minimisation problems as maximisation problems, since the same values x^* also satisfy

$$\begin{aligned} x^* = \arg \min_{x\in\mathbb{R}^n} &\ -c^{\mathrm{T}}x \\ \text{s.t. } &Ax \le b,\ x \ge 0,\ x \in \mathbb{Z}^n. \end{aligned}$$

In the same vein, any problem with *equality constraints*

$$\begin{aligned} \max_{x\in\mathbb{R}^n} &\ c^{\mathrm{T}}x \\ \text{s.t. } &Ax = b,\ x \ge 0,\ x \in \mathbb{Z}^n \end{aligned}$$

can be reformulated as a problem with *inequality constraints*

$$\begin{aligned} \max_{x\in\mathbb{R}^n} &\ c^{\mathrm{T}}x \\ \text{s.t. } &\begin{bmatrix} A \\ -A \end{bmatrix} x \le \begin{bmatrix} b \\ -b \end{bmatrix},\ x \ge 0,\ x \in \mathbb{Z}^n, \end{aligned}$$

since $Ax = b$ if and only if $Ax \le b$ and $-Ax \le -b$.

A type of IP that often occurs uses binary variables to indicate whether or not a particular feature is switched on. For example, if we have to choose 2 out of 4 objects worth $c_1, \dots, c_4$ respectively and aim to maximise the total value, we would have to solve the following problem,

$$\begin{aligned} \max_{x\in\mathbb{R}^4} &\ c^{\mathrm{T}}x \\ \text{subject to } &x_1 + x_2 + x_3 + x_4 = 2, \\ &0 \le x_i \le 1\ (i = 1, \dots, 4),\ x \in \mathbb{Z}^n. \end{aligned}$$

The constraints $0 \le x_i \le 1$, $x_i \in \mathbb{Z}$ force $x_i \in \mathbb{B} = \{0, 1\}$. Equivalently, we could replace this by $x_i(1 - x_i) = 0$, but this would no longer be a linear constraint and the problem could no longer be treated by the algorithms discussed in this course.

From this last example we learn that, while two mathematical formulations of an optimisation problem may be equivalent in terms of defining the same optimal values for the decision variables, the two formulations are generally not equivalent as far as their algorithmic treatment is concerned! This implies that finding a good formulation must be seen as part of the algorithmic solution of the problem, a theme we shall revisit over and over again in this course.

1.2 Complexity

- Often the coefficients of A, b, c are rational numbers, in which case the *problem size* $\mathscr{D}$ can be defined as the number of bits required to represent the problem data in reduced form.
- Generally, problems of larger size require more binary number operations to solve computationally on a computer, and the rate of growth in the required operations and memory as a function of problem size yields a notion of *problem complexity*.
- The best algorithms for linear optimisation problems *without* integrality constraints (LPs) have complexity bounded by $C \cdot \mathscr{D}^q$ for some fixed constants C, q, that is, such problems are *polynomial time solvable*.
- It is believed that there does not exist a polynomial time algorithm for solving general IP problems, and that the complexity of even the best algorithms grows exponentially in $\mathscr{D}$: IPs belong to the class of *NP hard optimisation problems*. That is, IP problems are hard to solve, due to the integrality constraints.
- Further details can be found in specialised courses or books on complexity theory. This theory will not be used in a substantial way in this course. Instead, we will focus on methods for solving IPs by iteratively solving much simpler sub-problems that increasingly better approximate the IP. Often the subproblems are LPs or special subclasses of IPs that are polynomial time solvable.

1.3 Introductory Examples

Example 1.1 (The Assignment Problem)**.** *A work force of n different workers are to carry out n different jobs. Each person can do any of the jobs in principle and must be assigned exactly one job. Assigning person i to job j incurs a cost c_{ij}. Find an assignment that minimises the total cost. See Figure 1 for an illustration.*

Decision variables: For $(i, j \in [1, n] := \{1, \dots, n\})$,

$$x_{ij} = \begin{cases} 1 & \text{if person } i \text{ assigned to carry out job } j, \\ 0 & \text{otherwise.} \end{cases}$$

Constraints:

- *Each person does exactly one job:*

$$\sum_{j=1}^{n} x_{ij} = 1 \qquad (i \in [1, n])$$

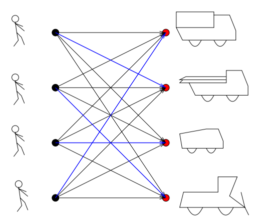


Figure 1: An illustration of the assignment problem.

- *Each job is done by exactly one person:*

$$\sum_{i=1}^{n} x_{ij} = 1 \qquad (j \in [1, n])$$

- *Variables are binary:*

$$x_{ij} \in \mathbb{B} := \{0, 1\}, \qquad (i, j \in [1, n]).$$

Objective function: the total cost $\sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij}$.

Model:

$$\begin{aligned}
\min_{x \in \mathbb{R}^{n \times n}} \quad & \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij} \\
\textit{s.t.} \quad & \sum_{j=1}^{n} x_{ij} = 1 \quad \textit{for } i = 1, \ldots, n, \\
& \sum_{i=1}^{n} x_{ij} = 1 \quad \textit{for } j = 1, \ldots, n, \\
& x_{ij} \in \mathbb{B} \quad \textit{for } i, j = 1, \ldots, n.
\end{aligned}$$

Example 1.2 (The 0-1 Knapsack Problem)**.** *A knapsack of volume b has to be packed with a selection of n items. Item i has volume a_i and value c_i, and we must pack the knapsack selecting a set of items of maximal total value. See Figure 2 for an illustration.*

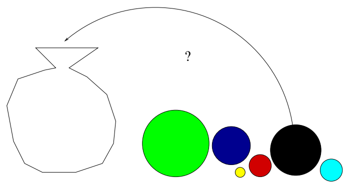


Figure 2: An illustration of the 0-1 knapsack problem.

Knapsack model:

$$\begin{aligned} \max \sum_{i=1}^{n} c_i x_i \\ s.t. \sum_{i=1}^{n} a_i x_i \leq b, \\ x \in \mathbb{B}^n, \end{aligned}$$

with decision variables are defined as follows,

$$x_i = \begin{cases} 1 & \textit{if item } i \textit{ is selected}, \\ 0 & \textit{otherwise}. \end{cases}$$

Example 1.3 (The Travelling Salesman Problem (TSP))**.** *A travelling salesman has to visit each of n cities exactly once and then return to the starting point. For each pair of cities $i, j \in [1, n]$ there is a direct air link from i to j. The directed graph (digraph) $G = (V, E)$ in which the vertices are the cities and the directed edges are the air links between them is assumed to be a complete graph. It takes c_{ij} hours to travel along edge ij from city i to city j. In which order to visit the cities so as to minimise the total travelling time? See Figure 3 for an illustration.*

Note: it may be the case that $c_{ij} \neq c_{ji}$. If $c_{ij} = c_{ji}$ for all $i, j \in [1, n]$, then we speak of the symmetric travelling salesman problem *(STSP), and (V, E) is considered an undirected graph.*

Formulation as a BIP:

- *Decision variables: for all $i, j \in [1, n]$,*

$$x_{ij} = \begin{cases} 1 & \textit{if the tour contains arc } (i, j), \\ 0 & \textit{otherwise}. \end{cases}$$

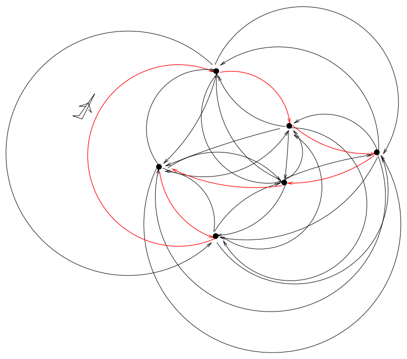


Figure 3: An illustration of the travelling salesman problem.

- *Constraints: the salesman leaves city i exactly once*

$$\sum_{j:j\neq i} x_{ij} = 1 \qquad (i = 1, \ldots, n)$$

and arrives in city j exactly once

$$\sum_{i:i\neq j} x_{ij} = 1 \qquad (j = 1, \ldots, n).$$

To eliminate solutions with subtours, introduce cut-set constraints*:*

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

See Figure 4 for an illustration of subtours.

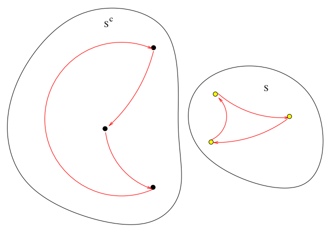


Figure 4: An illustration of subtours in the travelling salesman problem.

TSP model:

$$
\begin{aligned}
\min_x \ & \sum_{i=1}^n \sum_{j=1}^n c_{ij} x_{ij} \\
\text{s.t.} \ & \sum_{j:j\neq i} x_{ij} = 1 \quad (i \in [1,n]), \\
& \sum_{i:i\neq j} x_{ij} = 1 \quad (j \in [1,n]), \\
& \sum_{i\in S} \sum_{j\notin S} x_{ij} \geq 1 \quad (S \subset V,\ S \neq \emptyset), \\
& x \in \mathbb{B}^{n\times n}.
\end{aligned}
$$

Example 1.4 (Uncapacitated Facility Location (UFL))**.** *Potential sites $N = \{1,\dots,n\}$, clients $M = \{1,\dots,m\}$ for building a new distribution warehouse have been identified. It costs c_{ij} for satisfying all of client i's orders from warehouse j. Opening depot j incurs a fixed cost $f_j > 0$. Decide which warehouse to open and how to service clients at minimal cost. See Figure 5 for an illustration.*

Decision variables:

- *For each pair $(i,j) \in M \times N$ let $x_{ij} \in [0,1]$ be the proportion of the demand of of client i satisfied from depot j.*
- *Fixed costs: for each $j \in N$, let $y_j = 1$ if depot j is used, and $y_j = 0$ otherwise.*

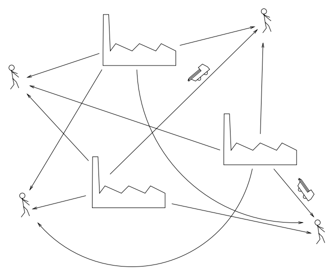


Figure 5: Illustration of the UFL.

Objective: minimise the total cost

$$\sum_{i\in M}\sum_{j\in N} c_{ij}x_{ij} + \sum_{j\in N} f_j y_j.$$

Constraints:

- *100% of client i's demand must be satisfied*

$$\sum_{j=1}^{n} x_{ij} = 1 \quad \textit{for } i = 1,\dots,m.$$

- $0 \le x_{ij} \le 1$ *for all* ij. *Note that the constraints* $x_{ij} \le 1$ *are superfluous, as they are implied by* $\sum_{j=1}^{n} x_{ij} = 1$.
- $y_j \in \mathbb{B}$, *with* $y_j = 1$ *iff* $\exists\, i \in M$ *s.t.* $x_{ij} > 0$*:*

$$\sum_{i=1}^{m} x_{ij} \le m y_j \quad \textit{for } j = 1,\dots,n.$$

This is an example of a so-called big M constraint.

UFL model (mixed integer programming problem):

$$\begin{aligned}
\min_{(x,y)\in\mathbb{R}^{m\times n+n}} \quad & \sum_{i\in M}\sum_{j\in N} c_{ij}x_{ij} + \sum_{j\in N} f_j y_j \\
\text{s.t.} \quad & \sum_{j\in N} x_{ij} = 1 \quad (i \in M), \\
& \sum_{i\in M} x_{ij} \leq m y_j \quad (j \in N), \\
& x \geq 0, \\
& y \in \mathbb{B}^n.
\end{aligned}$$

Example 1.5 (Discrete Alternatives (Disjunctions))**.**

$$\begin{aligned}
\min_{x\in\mathbb{R}^n} \quad & c^{\mathrm{T}}x \\
\text{s.t.} \quad & 0 \leq x \leq u \quad \textit{and (22) or (23) holds}, \\
& a_1^{\mathrm{T}}x \leq b_1 && (1) \\
& a_2^{\mathrm{T}}x \leq b_2, && (2)
\end{aligned}$$

where a_i are vectors and b_i scalars $(i = 1, 2)$. The "or" is inclusive, i.e., at least one *of conditions (22) and (23) must hold. See Figure 6 for an illustration.*

Modelling such problems as a MIPs: "Big M formulation"

- *Extra decision variables for which of (22),(23) to impose:*

$$y_1 = \begin{cases} 1 & \text{if (22) is imposed,} \\ 0 & \text{if (22) is not imposed,} \end{cases}$$

$$y_2 = \begin{cases} 1 & \text{if (23) is imposed,} \\ 0 & \text{if (23) is not imposed.} \end{cases}$$

  Note that even if a condition is not imposed, it may still hold, but when it is imposed, it must *hold.*

- *Let $M \geq \max\{\max_{0\leq x\leq u} a_i^{\mathrm{T}}x - b_i : i = 1, 2\}$, a bound that can easily be computed via linear programming (see Lecture 2).*

Alternative disjunction model:

$$\begin{aligned}
\min_{(x,y)\in\mathbb{R}^{n+2}} \quad & c^{\mathrm{T}}x \\
\text{s.t.} \quad & a_i^{\mathrm{T}}x - b_i \leq M(1 - y_i) \quad (i = 1, 2), \\
& y_1 + y_2 = 1, \\
& 0 \leq x \leq u \\
& y \in \mathbb{B}^2.
\end{aligned}$$

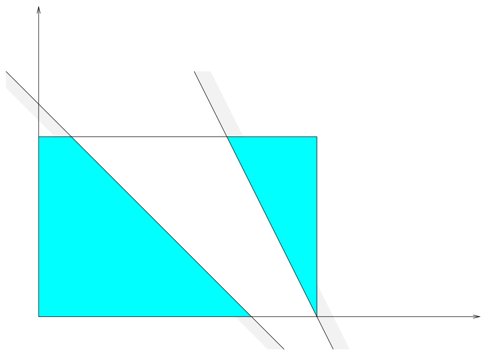


Figure 6: Illustration of the problem of discrete alternatives.

We make the following Observations:

- *If $y_1 = 1$, then $a_1^{\mathrm{T}} x - b_1 \leq M(1 - y_i) = 0$, and hence (22) is imposed.*
- *The case $y_1 = 0$ is more interesting: We have $a_1^{\mathrm{T}} x - b_1 \leq M$, which is a superfluous constraint, as it is already imposed as a consequence of $0 \leq x \leq u$. M has to be chosen large enough to guarantee that*

  $$\{x : 0 \leq x \leq u\} \cap \{x : A_1^{\mathrm{T}} x - b_1 \leq M\} = \{x : 0 \leq x \leq u\}$$

  so we don't lose any admissible solutions!

2 Lecture 2: Linear Programming Primal

2.1 LP Relaxation

An important class of optimisation problems are *linear programming* problems (LPs), which look just like IPs but without integrality constraints,

$$\begin{aligned} &\max_x c^{\mathrm{T}} x \\ \text{s.t.}\quad & Ax \leq b, \\ & x \geq 0. \end{aligned}$$

We will see that LPs play an important role in algorithms designed to solve general IPs through the concept of of *LP relaxation*:

Consider the IP problem

$$\begin{aligned} \text{(IP)}\quad z^* = &\max_x c^{\mathrm{T}} x \\ \text{s.t.}\quad & Ax \leq b, \\ & x \in \mathbb{Z}^n_+. \end{aligned}$$

If we give up on the integrality constraints $x_i \in \mathbb{Z}$, we obtain an LP,

$$\begin{aligned} \text{(LP)}\quad \bar{z} = &\max_x c^{\mathrm{T}} x \\ \text{s.t.}\quad & Ax \leq b, \\ & x \geq 0. \end{aligned}$$

Giving up on the integrality constraints has two effects on the feasible set $\mathscr{F}$ (the set of decision vectors x that satisfy the constraints of the problem)

- $\mathscr{F}$ becomes larger,
- $\mathscr{F}$ becomes convex.

See Figure 7 for an illustration of this effect.

Proposition 2.1 (Relaxation implies dual bound).
The consequence of the first effect is that $\bar{z} \geq z^*$.

Proof. If the optimal objective value z^* of (IP) is achieved at the point x^*, then x^* is feasible for (IP), and hence it is also feasible for (LP). Therefore,

$$\bar{z} \geq c^{\mathrm{T}} x = z^*.$$

□

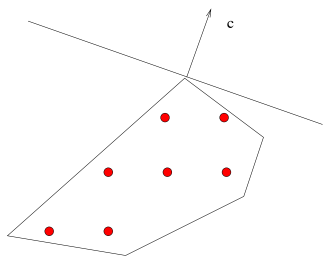


Figure 7: Illustration of LP relaxation.

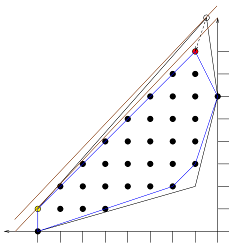


Figure 8: Rounded solutions may be far from optimal ones.

As we shall learn, the consequence of the second effect is that it is much easier to solve the problem (LP) than (IP).

A first idea for solving IPs is to solving the LP relaxation and round the optimal values of the decision variables to the nearest feasible integer valued feasible solution. While this occasionally works, it is not always a good idea:

- Rounding may be non-trivial, e.g., when the LP relaxation of a binary program takes an optimal solution x^* with many values near 0.5.
- The rounded solution may be far from optimal. See Figure 8 for an illustration of this effect.

2.2 Introductory Example

We will now discuss an algorithm for solving general linear programming problems.

Example 2.2 (Simplex in dictionary form)**.** *Consider the LP instance*

$$\begin{aligned} z = \max_x\, & 5x_1 + 4x_2 + 3x_3 \\ \textit{s.t.} \quad & 2x_1 + 3x_2 + x_3 \leq 5 \\ & 4x_1 + x_2 + 2x_3 \leq 11 \\ & 3x_1 + 4x_2 + 2x_3 \leq 8 \\ & x_1, x_2, x_3 \geq 0. \end{aligned}$$

Preliminary step I: introduce *slack variables* $x_4, x_5, x_6 \geq 0$ to reformulate inequality constraints as a system of linear equations,

$$\begin{aligned} z = \max\, & 5x_1 + 4x_2 + 3x_3 + 0x_4 + 0x_5 + 0x_6 \\ \text{s.t.} \quad & 2x_1 + 3x_2 + x_3 + x_4 = 5 \\ & 4x_1 + x_2 + 2x_3 + x_5 = 11 \\ & 3x_1 + 4x_2 + 2x_3 + x_6 = 8 \\ & x_1, x_2, x_3, x_4, x_5, x_6 \geq 0. \end{aligned}$$

Preliminary step II: express in *dictionary form*

$$\max z \quad \text{s.t.} \quad x_1, \dots, x_6 \geq 0,$$

and where the variables are linked via the linear system

$$\begin{aligned} x_4 &= 5 - 2x_1 - 3x_2 - x_3 \\ x_5 &= 11 - 4x_1 - x_2 - 2x_3 \\ x_6 &= 8 - 3x_1 - 4x_2 - 2x_3 \\ z &= 0 + 5x_1 + 4x_2 + 3x_3. \end{aligned}$$

Step 0: $x_1, x_2, x_3 = 0$, $x_4 = 5$, $x_5 = 11$, $x_6 = 8$ is an initial feasible solution. x_1, x_2, x_3 are called the *nonbasic variables* and x_4, x_5, x_6 *basic variables*. Note that basic variables are expressed in terms of nonbasic ones!

$$\begin{aligned} x_4 &= 5 - 2x_1 - 3x_2 - x_3 \\ x_5 &= 11 - 4x_1 - x_2 - 2x_3 \\ x_6 &= 8 - 3x_1 - 4x_2 - 2x_3 \\ z &= 0 + 5x_1 + 4x_2 + 3x_3. \end{aligned}$$

Step 1: We note that as long as x_1 is increased by at most

$$\frac{5}{2} = \min\left(\frac{5}{2}, \frac{11}{4}, \frac{8}{3}\right),$$

all x_i remain nonnegative, but z increases. Setting $x_1 = 5/2$ and substituting into the dictionary, we find $x_2, x_3, x_4 = 0$, $x_5 = 1$, $x_6 = 1/2$, $z = 25/2$ as an improved feasible solution. We call x_1 the *pivot* of the iteration.

$$\begin{aligned} x_4 &= 5 - 2x_1 - 3x_2 - x_3 \\ x_5 &= 11 - 4x_1 - x_2 - 2x_3 \\ x_6 &= 8 - 3x_1 - 4x_2 - 2x_3 \\ z &= 0 + 5x_1 + 4x_2 + 3x_3. \end{aligned}$$

We can now express the variables x_1, x_5, x_6, z in terms of the *new nonbasic variables* x_2, x_3, x_4 (those currently set to zero) to obtain a new dictionary. To do this, use line 1 of the dictionary to express x_1 in terms of x_2, x_3, x_4,

$$x_1 = \frac{1}{2}\big(5 - 3x_2 - x_3 - x_4\big)$$

and substitute the right hand side for x_1 in the remaining equations.

The new dictionary then looks as follows,

$$\begin{aligned} x_1 &= \frac{5}{2} - \frac{3}{2}x_2 - \frac{1}{2}x_3 - \frac{1}{2}x_4 && (3) \\ x_5 &= 1 + 5x_2 + 2x_4 && (4) \\ x_6 &= \frac{1}{2} + \frac{1}{2}x_2 - \frac{1}{2}x_3 + \frac{3}{2}x_4 && (5) \\ z &= \frac{25}{2} - \frac{7}{2}x_2 + \frac{1}{2}x_3 - \frac{5}{2}x_4. && (6) \end{aligned}$$

Of course, we are still solving

$$\max z \quad \text{s.t.} \quad x_1, \dots, x_6 \geq 0,$$

subject to the relationships (3)–(6) holding between the variables, and the new LP instance is equivalent to the old one. However, a better feasible solution can be read off the new dictionary by setting the nonbasic variables to zero!

$$\begin{aligned} x_1 &= \frac{5}{2} - \frac{3}{2}x_2 - \frac{1}{2}x_3 - \frac{1}{2}x_4 \\ x_5 &= 1 + 5x_2 + 2x_4 \\ x_6 &= \frac{1}{2} + \frac{1}{2}x_2 - \frac{1}{2}x_3 + \frac{3}{2}x_4 \\ z &= \frac{25}{2} - \frac{7}{2}x_2 + \frac{1}{2}x_3 - \frac{5}{2}x_4. \end{aligned}$$

Step 2: We continue in the same vein: increasing the value of x_2 or x_4 is useless, as this would decrease the objective value z. Thus, x_3 is our pivot, and we can increase its value up to

$$1 = \min\big(5, +\infty, 1\big),$$

leading to the improved solution $x_2, x_4, x_6 = 0$, $x_1 = 2$, $x_3 = 1$, $x_5 = 1$, $z = 13$ and the dictionary corresponding to x_2, x_4, x_6 as nonbasic variables:

$$\begin{aligned} x_3 &= 1 + x_2 + 3x_4 - 2x_6 \\ x_1 &= 2 - 2x_2 - 2x_4 + x_6 \\ x_5 &= 1 + 5x_2 + 2x_4 \\ z &= 13 - 3x_2 - x_4 - x_6. \end{aligned}$$

At this point we can stop the algorithm for the following reasons:

- from the last line of the dictionary we see that for any strictly positive value of x_2, x_4 or x_6 the objective value z is necessarily strictly smaller than 13,
- and from the other lines of the dictionary we see that as soon as the values of x_2, x_4 and x_6 are fixed, the values of x_3, x_1 and x_5 are fixed too.
- Thus, the last dictionary yields a certificate of optimality for the identified solution.

2.3 The Simplex Method

Let us now try to understand how the dictionary

$$\begin{aligned} x_3 &= 1 + x_2 + 3x_4 - 2x_6 \\ x_1 &= 2 - 2x_2 - 2x_4 + x_6 \\ x_5 &= 1 + 5x_2 + 2x_4 \\ z &= 13 - 3x_2 - x_4 - x_6, \end{aligned} \tag{7}$$

(which was obtained after two pivoting steps) could have been obtained directly from the input data of the original LP instance

$$\begin{aligned} \text{(LPI)} \quad \max\ & 5x_1 + 4x_2 + 3x_3 + 0x_4 + 0x_5 + 0x_6 \\ \text{s.t. } & 2x_1 + 3x_2 + x_3 + x_4 = 5 \\ & 4x_1 + x_2 + 2x_3 + x_5 = 11 \\ & 3x_1 + 4x_2 + 2x_3 + x_6 = 8 \\ & x_1, x_2, x_3, x_4, x_5, x_6 \geq 0 \end{aligned}$$

if we had been given the relevant basic variables:

The constraints of (LPI) imply a functional dependence between the nonnegative decision variables x_i, expressed by the linear system

$$Ax = b, \tag{8}$$

where

$$A = \begin{bmatrix} 2 & 3 & 1 & 1 & 0 & 0 \\ 4 & 1 & 2 & 0 & 1 & 0 \\ 3 & 4 & 2 & 0 & 0 & 1 \end{bmatrix}, \quad b = \begin{bmatrix} 5 \\ 11 \\ 8 \end{bmatrix}.$$

The basic variables of dictionary (7) are x_3, x_1, x_5. Writing

$$x_B := \begin{bmatrix} x_3 & x_1 & x_5 \end{bmatrix}^{\mathrm{T}}, \qquad x_N := \begin{bmatrix} x_2 & x_4 & x_6 \end{bmatrix}^{\mathrm{T}}$$

$$A_B := \begin{bmatrix} 1 & 2 & 0 \\ 2 & 4 & 1 \\ 2 & 3 & 0 \end{bmatrix}, \qquad A_N := \begin{bmatrix} 3 & 1 & 0 \\ 1 & 0 & 0 \\ 4 & 0 & 1 \end{bmatrix}$$

(8) can be written as

$$A_B\, x_B + A_N\, x_N = b.$$

Solving for the basic variables x_B, we obtain

$$x_B = A_B^{-1} \left(b - A_N x_N\right). \tag{9}$$

Likewise, the objective function can be written as

$$z = c_B^{\mathrm{T}}\, x_B + c_N^{\mathrm{T}}\, x_N,$$

where

$$c_B = \begin{bmatrix} 3 & 5 & 0 \end{bmatrix}^{\mathrm{T}}, \quad c_N = \begin{bmatrix} 4 & 0 & 0 \end{bmatrix}^{\mathrm{T}},$$

and substituting from (9), we find

$$z = c_B^{\mathrm{T}} A_B^{-1} b + \left(c_N^{\mathrm{T}} - c_B^{\mathrm{T}} A_B^{-1} A_N\right) x_N.$$

Dictionary (7) is now just the system of equations

$$\begin{aligned} x_B &= A_B^{-1} b - A_B^{-1} A_N x_N, \\ z &= c_B^{\mathrm{T}} A_B^{-1} b + \left(c_N^{\mathrm{T}} - c_B^{\mathrm{T}} A_B^{-1} A_N\right) x_N. \end{aligned}$$

Definition 2.3 (Dictionary). *A* dictionary *of the LP problem (P)*

$$\max_x \{c^{\mathrm{T}} x : \ Ax = b,\ x \geq 0\}$$

is a system of equations

$$\begin{aligned} x_B &= A_B^{-1} b - A_B^{-1} A_N x_N, \\ z &= c_B^{\mathrm{T}} A_B^{-1} b + \left(c_N^{\mathrm{T}} - c_B^{\mathrm{T}} A_B^{-1} A_N\right) x_N, \end{aligned}$$

equivalent to

$$\begin{aligned} Ax &= b, \\ z &= c^{\mathrm{T}} x, \end{aligned}$$

where up to column permutation $A = [\,A_B\ A_N\,]$ and $x = [\,x_B^{\mathrm{T}}\ x_N^{\mathrm{T}}\,]^{\mathrm{T}}$ is a block decomposition such that A_B is nonsingular.

A dictionary is called feasible *if $A_B^{-1}b \geq 0$, so that $x = (x_B, x_N) = (A_B^{-1}b, 0)$ is feasible (but generally suboptimal). The point (x_B, x_N) is then called a* basic feasible solution*.*

Algorithm 2.4 (Simplex Method for the general LP instance (P))**.**

```
// initialisation
choose a basic feasible solution $(x_B, x_N)$;
while $c_N - A_N^{\mathrm{T}} A_B^{-T} c_B > 0$ do
    $i := \min\{\ell \in N : c_\ell > A_\ell^{\mathrm{T}} A_B^{-T} c_B\}$; // $A_\ell$ is $\ell$-th column of $A$
    if $A_B^{-1} A_i \leq 0$ then
        return "(P) unbounded"; // (objective $\to \infty$)
    else
        $AM := \arg\min_{k \in B} \{(A_B^{-1}b)_k / (A_B^{-1}A_i)_k : k \in B, (A_B^{-1}A_i)_k > 0\}$;
        $j := \min_{\ell \in AM} \ell$;
        $N \leftarrow N \cup \{j\} \setminus \{i\}$;
        $B \leftarrow B \cup \{i\} \setminus \{j\}$;
    end
end
return $(x_B, x_N) = (A_B^{-1}b, 0)$ "optimal basic solution";
```

Comments:

- The specific choice of i and j in Steps i) and ii) provably avoid that the same sets (B, N) appear twice in the course of the algorithm, and hence guarantee finite termination. This is called Bland's Rule, the proof of which we omit.
- In an efficient implementation the full dictionary is never computed, as it suffices to compute the vectors $c_N - A_N^{\mathrm{T}} A_B^{-T} c_B$ and $A_B^{-1} A_i$ (which computationally amounts to solving two linear systems, one matrix-vector multiplication and one vector addition), as well as $A_B^{-1}b$ (a clever implementation does not require an extra linear systems solve).

2.4 Tableau Format of the Simplex Method

A tableau is a representation of the problem data obtained by moving all variables x_i of a dictionary to one side and constants to the other. For example, the dictionary

$$\begin{aligned} x_4 &= 5 - 2x_1 - 3x_2 - x_3 \\ x_5 &= 11 - 4x_1 - x_2 - 2x_3 \\ x_6 &= 8 - 3x_1 - 4x_2 - 2x_3 \\ z &= 0 + 5x_1 + 4x_2 + 3x_3, \end{aligned}$$

corresponds to the tableau

$$\begin{array}{cccccc||c}
2 & 3 & 1 & 1 & 0 & 0 & 5 \\
4 & 1 & 2 & 0 & 1 & 0 & 11 \\
3 & 4 & 2 & 0 & 0 & 1 & 8 \\
\hline\hline
5 & 4 & 3 & 0 & 0 & 0 & 0.
\end{array}$$

Basic variables can be identified via the appearance of an identity submatrix, and an optimal tableau is characterised by the appearance of all non-positive entries on the l.h.s. of the last line.

Applying the same procedure to the second dictionary

$$\begin{aligned}
x_1 &= \frac{5}{2} - \frac{3}{2}x_2 - \frac{1}{2}x_3 - \frac{1}{2}x_4 \\
x_5 &= 1 + 5x_2 + 2x_4 \\
x_6 &= \frac{1}{2} + \frac{1}{2}x_2 - \frac{1}{2}x_3 + \frac{3}{2}x_4 \\
z &= \frac{25}{2} - \frac{7}{2}x_2 + \frac{1}{2}x_3 - \frac{5}{2}x_4,
\end{aligned}$$

we obtain the following tableau,

$$\begin{array}{cccccc||c}
1 & 1.5 & 0.5 & 0.5 & 0 & 0 & 2.5 \\
0 & -5 & 0 & -2 & 1 & 0 & 1 \\
0 & -0.5 & 0.5 & -2.5 & 0 & 1 & 0.5 \\
\hline\hline
0 & -3.5 & 0.5 & -2.5 & 0 & 0 & -12.5.
\end{array}$$

Here is how the first tableau can be directly transformed into the second one:

1. Among the columns on the left, identify one whose last entry is positive.

$$\begin{array}{|c|ccccc||c}
2 & 3 & 1 & 1 & 0 & 0 & 5 \\
4 & 1 & 2 & 0 & 1 & 0 & 11 \\
3 & 4 & 2 & 0 & 0 & 1 & 8 \\
\hline\hline
5 & 4 & 3 & 0 & 0 & 0 & 0.
\end{array}$$

   We call this column the *pivot column*.

$$\begin{array}{|c|ccccc||c}
2 & 3 & 1 & 1 & 0 & 0 & 5 \\
4 & 1 & 2 & 0 & 1 & 0 & 11 \\
3 & 4 & 2 & 0 & 0 & 1 & 8 \\
\hline\hline
5 & 4 & 3 & 0 & 0 & 0 & 0.
\end{array}$$

2. For each row (apart from the last) of the pivot column with positive coefficient t, look up the corresponding coefficient u on the r.h.s.

- If no such row exists, the problem is unbounded.
- If such rows exist, pick the one for which u/t is smallest and call it the *pivot row*. In this example, $t = 2$ and $u = 5$.

2	3	1	1	0	0	5
4	1	2	0	1	0	11
3	4	2	0	0	1	8
5	4	3	0	0	0	0.

3. Divide the pivot row by t,

1	1.5	0.5	0.5	0	0	2.5
4	1	2	0	1	0	11
3	4	2	0	0	1	8
5	4	3	0	0	0	0.

4. For all other rows i of the tableau, subtract the "rescaled" pivot row t_i times, where t_i is the row-i entry of the pivot colum,

1	1.5	0.5	0.5	0	0	2.5
0	−5	0	−2	1	0	1
0	−0.5	0.5	−2.5	0	1	0.5
0	−3.5	0.5	−2.5	0	0	−12.5.

In the new tableau there once again appears a permuted identity matrix that identifies the new basic variables.

2.5 Phase I of the Simplex Algorithm

A problem we neglected so far is that the simplex algorithm needs to be given a basic feasible solution as a starting point to be able to run, and finding such a solution is often a non-trivial problem in itself. Luckily, we can first solve the auxiliary LP instance

$$\text{(AUX)} \qquad \max_{y=(s,t,x)\in\mathbb{R}^{m+m+n}} \ -\sum_{i=1}^{m}(s_i+t_i)$$
$$\text{s.t.} \ \begin{bmatrix} \mathrm{I} & -\mathrm{I} & A \end{bmatrix} \begin{bmatrix} s \\ t \\ x \end{bmatrix} = b,$$
$$s, t, x \geq 0.$$

A basic feasible solution for (AUX) is readily given by

$$B = \{i : 1 \leq i \leq m, \, b_i \geq 0\} \cup \{m+i : 1 \leq i \leq m, \, b_i < 0\},$$

so that $y_B = |b|$ consists of components of s and t.

Further, the optimal basic solution $y^* = (s^*, t^*, x^*)$ of the auxiliary problem satisfies exactly one of the following two conditions:

i) Either $s^*, t^* = 0$ and then w.l.o.g. all the y-indices corresponding to s and t can be assumed to have been pivoted into the set of nonbasic variables y^*_N, so that
$$(x_B, x_N) = (\pi_x^B y^*_B, \pi_x^N y^*_N)$$
is a basic feasible solution for (P), where the maps $\pi_x^{\cdot}$ are the projections of the relevant parts of y onto their x-components.

ii) Or else, $\sum_{i=1}^m (s_i + t_i) > 0$ proves that the set of feasible solutions for (P) is empty, since any feasible solution x of (P) together with $s, t = 0$ would improve on the optimal objective value.

3 Lecture 3: Linear Programming Duality

3.1 Bounding LPs

Let us again consider the LP instance we studied previously,

$$\begin{aligned}
\text{(P)} \quad & \max\ 5x_1 + 4x_2 + 3x_3 \\
& \text{s.t. } 2x_1 + 3x_2 + x_3 \le 5 \\
& \quad 4x_1 + x_2 + 2x_3 \le 11 \\
& \quad 3x_1 + 4x_2 + 2x_3 \le 8 \\
& \quad x_1, x_2, x_3 \ge 0.
\end{aligned}$$

We saw that the optimal value is 13.

In integer programming, instead of solving an LP relaxation to optimality one is often interested in finding merely upper and lower bounds on the optimal value. A lower bound is provided by any feasible solution. For example, $x_1, x_2 = 1$, $x_3 = 0$ is feasible with objective value 9.

How can we obtain upper bounds?

- Multiplying the first constraint by 3 we obtain

$$6x_1 + 9x_2 + 3x_3 \le 15,$$

  and since $x_1, x_2, x_3 \ge 0$, this yields an upper bound on the objective function:

$$z = 5x_1 + 4x_2 + 3x_3 \le 6x_1 + 9x_2 + 3x_3 \le 15,$$

- Likewise, taking the sum of the first two constraints yields the valid upper bound

$$z = 5x_1 + 4x_2 + 3x_3 \le 6x_1 + 4x_2 + 3x_3 \le 16.$$

- More generally, such bounds can be obtained from any sum of positive multiples of the constraints for which the resulting coefficients are no smaller than the corresponding coefficients of the objective function:

$$\begin{bmatrix} 5 & 4 & 3 \end{bmatrix} \le \begin{bmatrix} y_1 & y_2 & y_3 \end{bmatrix} \begin{bmatrix} 2 & 3 & 1 \\ 4 & 1 & 2 \\ 3 & 4 & 2 \end{bmatrix}, \quad y_1, y_2, y_3 \ge 0$$

$$\Rightarrow z \le \begin{bmatrix} y_1 & y_2 & y_3 \end{bmatrix} \begin{bmatrix} 5 \\ 11 \\ 8 \end{bmatrix}.$$

The best such upper bound is obtained by solving the LP instance

$$\begin{aligned}\text{(D)}\quad & \min_y 5y_1 + 11y_2 + 8y_3 \\ & \text{s.t. } 2y_1 + 4y_2 + 3y_3 \geq 5, \\ & \quad 3y_1 + y_2 + 4y_3 \geq 4, \\ & \quad y_1 + 2y_2 + 2y_3 \geq 3, \\ & \quad y_1, y_2, y_3 \geq 0.\end{aligned}$$

This is called the *dual* of the LP instance (P), the latter being called the *primal*.

More generally, an LP of the form

$$\begin{aligned}\text{(P)}\quad z^* = & \max_x c^{\mathrm{T}} x \\ & \text{s.t. } Ax \leq b \\ & \quad x \geq 0\end{aligned}$$

is associated with a dual

$$\begin{aligned}\text{(D)}\quad w^* = & \min_y b^{\mathrm{T}} y \\ & \text{s.t. } A^{\mathrm{T}} y \geq c \\ & \quad y \geq 0,\end{aligned}$$

since

$$y^{\mathrm{T}} A \geq c^{\mathrm{T}} \overset{x \geq 0}{\Longrightarrow} y^{\mathrm{T}} A x \geq c^{\mathrm{T}} x \overset{y \geq 0,\, Ax \leq b}{\Longrightarrow} y^{\mathrm{T}} b \geq c^{\mathrm{T}} x.$$

This allows us to formulate a dual problem for any LP, by first reformulating it in canonical form. For example,

$$\begin{aligned}\text{(P)}\quad z^* = & \max_x c^{\mathrm{T}} x \\ & \text{s.t. } Ax \leq b.\end{aligned}$$

Writing $x = x^1 - x^2$ with $x^1,\, x^2 \geq 0$, this is equivalent to

$$\begin{aligned}\text{(P}'\text{)}\quad z^* = & \max_{x^1, x^2} [c^{\mathrm{T}}, -c^{\mathrm{T}}] \begin{bmatrix} x^1 \\ x^2 \end{bmatrix} \\ & \text{s.t. } [A,\, -A] \begin{bmatrix} x^1 \\ x^2 \end{bmatrix} \leq b, \\ & \quad x^1,\, x^2 \geq 0.\end{aligned}$$

The latter has the dual

$$\begin{aligned}\text{(D}'\text{)}\quad w^* = & \min_y b^{\mathrm{T}} y \\ & \text{s.t. } A^{\mathrm{T}} y \geq c, \\ & \quad -A^{\mathrm{T}} y \geq -c, \\ & \quad y \geq 0,\end{aligned}$$

which in turn is equivalent to

$$\begin{aligned}\text{(D)}\quad w^* = \min_y\ & b^{\mathrm{T}}y\\ \text{s.t. } & A^{\mathrm{T}}y = c,\\ & y \geq 0.\end{aligned}$$

3.2 Weak Duality

To analyse the relationship between the primal-dual pair (P), (D), we will henceforth consider LPs in the following *standard form* into which any LP may be cast under an appropriate reformulation,

$$\begin{aligned}\text{(P)}\quad \max_x\ & \sum_{j=1}^{n} c_j x_j\\ \text{s.t. } & \sum_{j=1}^{n} a_{ij}x_j \leq b_i,\ (i=1,\dots,m),\end{aligned}$$

$$\begin{aligned}\text{(D)}\quad \min_y\ & \sum_{i=1}^{m} y_i b_i\\ \text{s.t. } & \sum_{i=1}^{m} y_i a_{ij} = c_j,\ (j=1,\dots,n)\\ & y_i \geq 0,\ (i=1,\dots,m).\end{aligned}$$

Theorem 3.1 (Weak Duality Theorem)**.**

i) If x is primal feasible and y is dual feasible (feasible for (P), (D) respectively), then

$$\sum_{j=1}^{n} c_j x_j \leq \sum_{i=1}^{m} y_i b_i. \tag{10}$$

ii) If equality holds in (10), then x is primal optimal and y is dual optimal.

iii) If either (P) or (D) is unbounded, then the other programme is infeasible (has no feasible solutions).

Proof. i) By assumption, we have $\sum_{j=1}^{n} a_{ij}x_j \leq b_i$ for all i and $\sum_{i=1}^{m} y_i a_{ij} = c_j$, $y_i \geq 0$ for all j. Therefore,

$$\sum_{j=1}^{n} c_j x_j = \sum_{j=1}^{n}\left(\sum_{i=1}^{m} y_i a_{ij}\right)x_j = \sum_{i=1}^{m} y_i\left(\sum_{j=1}^{n} a_{ij}x_j\right) \leq \sum_{i=1}^{m} y_i b_i.$$

ii) and iii) are immediate consequences of i). □

Theorem 3.2 (Theorem of Alternatives for Linear Inequalities)**.** *Consider the following two systems of linear inequalities,*

$$\sum_{j=1}^{n} a_{ij}x_j \leq b_i, \quad (i = 1, \dots, m) \tag{11}$$

and

$$\begin{aligned} \sum_{i=1}^{m} y_i a_{ij} &= 0, \quad (j = 1, \dots, n) \\ y_i &\geq 0, \quad (i = 1, \dots, m) \\ \sum_{i=1}^{m} y_i b_i &< 0. \end{aligned} \tag{12}$$

Then system (11) *has a solution if and only if* (12) *has no solution.*

Proof. See problem sheets. □

3.3 Strong Duality

Theorem 3.3 (Strong Duality Theorem)**.**

i) If (P) and (D) both have feasible solutions, then they have optimal solutions x and y such that $\sum_{j=1}^{n} c_j x_j = \sum_{i=1}^{m} y_i b_i$.

ii) If either (P) or (D) is infeasible, then the other programme is either unbounded or infeasible.

Proof. Expanding each equality of the dual constraints as two inequalities, the claim of part i) may be written as the following system,

$$\begin{aligned} \sum_{j=1}^{n} a_{ij}x_j &\leq b_i,\ (i = 1, \dots, m) \text{ (primal feasibility)} \\ \sum_{i=1}^{m} y_i a_{ij} &\leq c_j,\ (j = 1, \dots, n) \text{ (dual feasibility)} \\ -\sum_{i=1}^{m} y_i a_{ij} &\leq -c_j,\ (j = 1, \dots, n) \text{ (dual feasibility)} \\ -y_i &\leq 0,\ (i = 1, \dots, m) \text{ (dual feasibility)} \\ -\sum_{j=1}^{n} c_j x_j + \sum_{i=1}^{m} b_i y_i &\leq 0 \text{ (optimality)} \end{aligned} \tag{13}$$

In matrix form (22) reads as

$$\text{(I)}\quad \begin{bmatrix} A & \\ & A^{\mathrm{T}} \\ & -A^{\mathrm{T}} \\ & -\mathrm{I} \\ -c^{\mathrm{T}} & b^{\mathrm{T}} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \le \begin{bmatrix} b \\ c \\ -c \\ 0 \\ 0 \end{bmatrix}$$

Writing this as $\tilde{A}\tilde{x} \le \tilde{b}$, the FTLI implies that (I) has a solution if and only if the following system does not have a solution, $\tilde{A}^{\mathrm{T}}\tilde{y} = 0$, $\tilde{y} \ge 0$, $\tilde{y}^{\mathrm{T}}\tilde{b} < 0$, or equivalently,

$$\text{(II)}\quad \begin{bmatrix} A^{\mathrm{T}} & & & & -c \\ & A & -A & -\mathrm{I} & b \end{bmatrix} \begin{bmatrix} u \\ v \\ w \\ s \\ \tau \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix},$$

$$b^{\mathrm{T}}u + c^{\mathrm{T}}(v-w) < 0,\ u, v, w, s, \tau \ge 0.$$

Writing $h = w - v$ the latter system can be rewritten as follows,

$$\begin{aligned} \sum_{i=1}^{m} u_i a_{ij} - c_j\tau &= 0,\ (j = 1, \dots, n) \\ -\sum_{j=1}^{n} a_{ij}h_j - s_i + b_i\tau &= 0,\ (i = 1, \dots, m) \\ u_i, s_i, \tau &\ge 0,\ (i = 1, \dots, m; j = 1, \dots, n) \\ \sum_{i=1}^{m} b_i u_i - \sum_{j=1}^{n} c_j h_j &< 0, \end{aligned} \tag{14}$$

To prove that (22) is feasible, we must show that assuming the feasibility of (23) leads to a contradiction.

Case 1: (23) has solution with $\tau > 0$. Let $x_j = \frac{1}{\tau}h_j\ \forall j$ and $y_i = \frac{1}{\tau}u_i\ \forall i$. Then x and y are primal and dual feasible, but

$$\sum_{i=1}^{m} b_i u_i + \sum_{j=1}^{n} c_j h_j < 0$$

implies $\sum_{i=1}^{m} y_i b_i < \sum_{j=1}^{n} c_j x_j$, violating weak duality.

Case 2: (23) has solution with $\tau = 0$ and $\sum_{i=1}^{m} u_i b_i < 0$. Take $\tilde{y}$ dual feasible. Then $y = \tilde{y} + \lambda u$ is dual feasible for all $\lambda \ge 0$, and

$$\sum_{i=1}^{m} y_i b_i = \sum_{i=1}^{m} \tilde{y}_i b_i + \lambda \sum_{i=1}^{m} u_i b_i$$

becomes arbitrarily negative for large λ. By weak duality, (P) is infeasible.

Case 3: (23) has solution with $\tau = 0$ and $\sum_{j=1}^{n} c_j h_j > 0$: similar construction to Case 2 with $x = \tilde{x} + \lambda h$ and $\tilde{x}$ primal feasible.

For parts ii) proceed similarly (see problem sheets). □

3.4 Linear Complementarity

Definition 3.4 (Complementarity)**.** *$x \in \mathbb{R}^n$ and $y \in \mathbb{R}^m$ are complementary solutions relative to (P) and (D) if*

$$y_i \left(b_i - \sum_{j=1}^{n} a_{ij} x_j \right) = 0, \quad (i = 1, \dots, m),$$

that is, either $y_i = 0$ or the primal constraint $\sum_{j=1}^{n} a_{ij} x_j \leq b_i$ is active (holds as equality).

Theorem 3.5 (Complementary Slackness)**.** *x and y are complementary solutions relative to (P) and (D) if and only if they are optimal solutions of (P) and (D).*

Proof. x and y are primal and dual feasible. Therefore,

$$\sum_{i=1}^{m} y_i \left(b_i - \sum_{j=1}^{n} a_{ij} x_j \right) = \sum_{i=1}^{m} y_i b_i - \sum_{j=1}^{n} \left(\sum_{i=1}^{m} y_i a_{ij} \right) x_j = \sum_{i=1}^{m} y_i b_i - \sum_{j=1}^{n} c_j x_j. \tag{15}$$

If x and y are complementary, then the l.h.s. of (15) equals zero, hence by weak duality, the r.h.s. shows that x and y are optimal.

If x and y are optimal, then by strong duality, the r.h.s. of (15) equals zero, and since the l.h.s. consists of non-negative summands, we have $y_i \left(b_i - \sum_{j=1}^{n} a_{ij} x_j \right)$ for all i. □

Definition 3.6 (Linear Complementarity)**.** *More generally, solutions (x, s) and (y, t) are complementary relative to the primal-dual pair of LP problems*

$$\begin{aligned}
\text{(P)} \quad z^* = \max_{(x,s)} \; & c^{\mathrm{T}} x + d^{\mathrm{T}} s & \qquad \text{(D)} \quad w^* = \min_{(y,t)} \; & a^{\mathrm{T}} y + b^{\mathrm{T}} t \\
\text{s.t.} \; & Ax + Cs \leq a, & \text{s.t.} \; & A^{\mathrm{T}} y + B^{\mathrm{T}} t \geq c \\
& Bx + Ds = b, & & C^{\mathrm{T}} y + D^{\mathrm{T}} t = d \\
& x \geq 0 & & y \geq 0.
\end{aligned}$$

if the following conditions are satisfied,

$$y_i\left(a_i - \sum_j a_{ij}x_j - \sum_k c_{ik}s_k\right) = 0, \quad \forall i,$$
$$\left(\sum_i y_i a_{ij} + \sum_\ell t_\ell b_{\ell j} - c_j\right) x_j = 0, \quad \forall j.$$

Corollary 3.6.1 (Complementarity and duality)**.** *Complementary solutions relative to a primal-dual pair of LPs remain complementary when the roles of primal and dual are interchanged.*

Proof. Follows from the the Complementary Slackness Theorem and the fact that the bi-dual equals the primal. □

Example 3.7 (Check optimality by complementarity)**.** *Consider the LP instance*

$$\begin{aligned}
\textit{(P)} \quad & \max 5x_1 + 4x_2 + 3x_3 \\
& \textit{s.t. } 2x_1 + 3x_2 + x_3 \leq 5 \\
& \quad 4x_1 + x_2 + 2x_3 \leq 11 \\
& \quad 3x_1 + 4x_2 + 2x_3 \leq 8 \\
& \quad x_1, x_2, x_3 \geq 0.
\end{aligned}$$

Suppose we are given the solution $x_1^* = 2$, $x_2^* = 0$, $x_3^* = 1$. How can we check if this solution is optimal for (P)? We verify the claim by constructing a dual optimal y^* via complementary slackness.

First we check that x^* is feasible for (P) by substitution into the constraint inequalities: If x^* is optimal, then by complementary slackness, $x_1^*, x_3^* > 0$ implies

$$2y_1^* + 4y_2^* + 3y_3^* = 5 \tag{16}$$
$$y_1^* + 2y_2^* + 2y_3^* = 3. \tag{17}$$

Furthermore, $4x_1^* + x_2^* + 2x_3^* = 10 < 11$ implies $y_2^* = 0$.

Substituting into (16) and (17), we obtain

$$\begin{aligned}
2y_1^* + 3y_3^* &= 5 \\
y_1^* + 2y_3^* &= 3
\end{aligned}$$

Solving this linear system, we find $y_1^* = 1$, $y_3^* = 1$.

By construction, y^* satisfies the constraints $A^{\mathrm{T}}y^* \geq c$, and we also see that $y^* \geq 0$. Furthermore,

$$c^{\mathrm{T}}x^* = 5 \cdot 2 + 4 \cdot 0 + 3 \cdot 1 = 13 = 5 \cdot 1 + 11 \cdot 0 + 8 \cdot 1 = b^{\mathrm{T}}y^*,$$

confirming the optimality of both x^* and y^*.

4 Lecture 4: Dual Simplex Algorithm, Alternative IP Formulations

4.1 The Dual Simplex Algorithm

Thanks to strong LP duality and strict complementarity, one could solve the dual of a given LP and derive the primal optimal solution from the dual optimal solution, if that is easier. Applying the simplex algorithm to the dual problem can be done directly in the primal tableau, and this is called the *dual simplex algorithm*.

Example 4.1 (Dual Simplex)**.**

$$\begin{aligned} \textit{(P)} \quad & \max_{x_1,x_2} -\frac{1}{4}x_1 - \frac{1}{4}x_2 \\ & \textit{s.t.}\ \frac{1}{6}x_1 - \frac{1}{6}x_2 \le 1, \\ & \quad \frac{1}{4}x_1 + \frac{1}{4}x_2 \le \frac{3}{2}, \\ & \quad -\frac{1}{4}x_1 - \frac{1}{4}x_2 \le -\frac{1}{2}, \\ & \quad x_1, x_2 \ge 0. \end{aligned}$$

Adding slack variables, we find the tableau

1/6	-1/6	1	0	0	1
1/4	1/4	0	1	0	3/2
-1/4	-1/4	0	0	1	-1/2
-1/4	-1/4	0	0	0	0

*The last line indicates that the tableau would be optimal if only the associated basic solution were feasible, which the r.h.s. shows it is not. We want to take pivots that keep the coefficients on the last line non-positive (*dual feasibility*) and make progress toward making the tableau feasible. To render x_5 non-negative, we must use row 3 as a pivot row in which to eliminate a different variable.*

To decide on which column to pivot, note that if the pivot row t were to read

$$\sum_{j=1}^{n} \bar{a}_{tj} x_j = \bar{b}_t$$

with $\bar{a}_{tj} \ge 0$ $(j = 1, \dots, n)$ and $\bar{b}_t < 0$, then no matter how $x \ge 0$ is chosen, constraint t could not be satisfied. In that case, we would have to conclude that the primal problem is infeasible.

Luckily, in our case, this is not so, and in pivoting on column h with $\bar{a}_{th} < 0$, the last row changes as follows,

$$\bar{c}_j \leftarrow \bar{c}_j - \frac{\bar{c}_h}{\bar{a}_{th}} \bar{a}_{tj}.$$

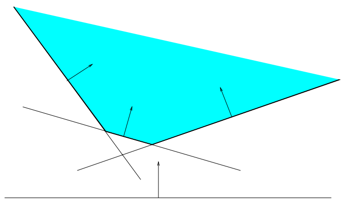


Figure 9: Illustration of a polyhedron.

To guarantee dual feasibility, we must not allow any $\bar{c}_j$ to become positive, that is,

$$\bar{c}_j - \frac{\bar{c}_h}{\bar{a}_{th}}\bar{a}_{tj} \le 0, \quad (j = 1, \dots, n) \quad \text{(not a problem if } \bar{a}_{tj} \ge 0\text{, since } \bar{c}_h \le 0 \text{ and } \bar{a}_{th} < 0\text{)}$$

$$\Leftrightarrow \frac{\bar{c}_j}{|\bar{a}_{tj}|} - \frac{\bar{c}_h}{|\bar{a}_{th}|} \le 0, \quad (j \in [1, n],\, \bar{a}_{tj} < 0)$$

$$\Leftrightarrow h \in \arg\max \left\{ \frac{\bar{c}_j}{|\bar{a}_{tj}|} : j \in [1, n],\, \bar{a}_{tj} < 0 \right\}.$$

For example in our case, $t = 3$, $h \in \{1, 2\}$. Eliminating x_1 in row 3 optimises the tableau,

$$\begin{array}{ccccc|c} 1/6 & -1/6 & 1 & 0 & 0 & 1 \\ 1/4 & 1/4 & 0 & 1 & 0 & 3/2 \\ 1 & 1 & 0 & 0 & -4 & 2 \\ \hline -1/4 & -1/4 & 0 & 0 & 0 & 0 \end{array} \rightarrow \begin{array}{ccccc|c} 0 & -1/3 & 1 & 0 & 2/3 & 2/3 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & -4 & 2 \\ \hline 0 & 0 & 0 & 0 & -1 & 1/2 \end{array}$$

4.2 Polyhedra and Polytopes

Definition 4.2 (Polyhedron). *A* polyhedron *is a set $\mathcal{P} \subset \mathbb{R}^n$ described as an intersection of finitely many affine half spaces*

$$\mathcal{P} = \left\{ x \in \mathbb{R}^n : \sum_{j=1}^{n} a_{ij} x_j \le b_i,\, (i = 1, \dots, m) \right\},$$

for some $A = (a_{ij}) \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$. See Figure 9 for an illustration of this concept.

Definition 4.3 (Polytope). *A* polytope *is a set $\mathcal{P}' \subset \mathbb{R}^n$ described as the convex hull of finitely many points*

$$\mathcal{P}' = \operatorname{conv}\left\{x^k : k \in [1, p]\right\} := \left\{ \sum_{k=1}^{p} \lambda_k x^k : \sum_{k=1}^{p} \lambda_k = 1,\, \lambda_k \ge 0,\, \forall k \right\}$$

for some $x^1, \dots, x^p \in \mathbb{R}^n$. See Figure 10 for an illustration of this concept.

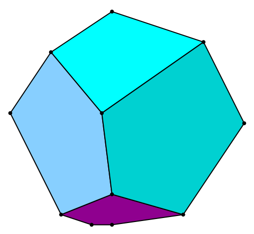


Figure 10: Illustration of a polytope.

Polytopes and polyhedra are essentially the same objects, except that polyhedra may extend to infinity and that the descriptions are duals of each other, one in terms of linear inequalities, and one by generators.

Theorem 4.4 (Weyl's Theorem)**.** *If $\mathcal{P}'$ is a polytope, then $\mathcal{P}'$ is also a polyhedron.*

Proof. $x \in \mathcal{P}'$ iff $\exists\, \lambda$ such that

$$\begin{aligned} x_j - \sum_{k=1}^{p} \lambda_k x_j^k &= 0, \quad (j = 1, \dots, n) \\ \sum_{k=1}^{p} \lambda_k &= 1, \\ \lambda_k &\geq 0, \quad (k = 1, \dots, p), \end{aligned}$$

where x_j and λ_k are variables but x_k^j are constants. Equivalently, the requirement can be written as a system of linear inequalities,

$$\begin{aligned} x_j - \sum_{k=1}^{p} \lambda_k x_j^k &\leq 0, \quad (j = 1, \dots, n) \\ -x_j + \sum_{k=1}^{p} \lambda_k x_j^k &\leq 0, \quad (j = 1, \dots, n) \\ \sum_{k=1}^{p} \lambda_k &\leq 1, \\ -\sum_{k=1}^{p} \lambda_k &\leq -1, \\ -\lambda_k &\leq 0, \quad (k = 1, \dots, p). \end{aligned}$$

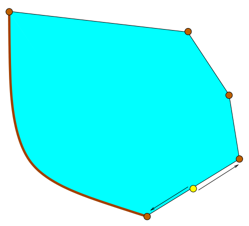


Figure 11: Extreme points of a convex set.

Now apply Fourier-Motzkin Elimination (see problem sheet) to eliminate all the variables λ_k. This yields a new system of inequalities

$$\sum_{j=1}^{n} a_{ij}x_j + \sum_{k=1}^{N} 0 \cdot \lambda_k \leq b_i, \quad (i = 1, \dots, m)$$

for some $A = (a_{ij})$, b and m, each of which is a positive linear combination of inequalities of the original system. By the properties of Fourier-Motzkin elimination, $\exists\, \lambda$ such that (x, λ) satisfy the orginal system iff x satisfies the new system, which now doesn't involve λ. Hence,

$$\mathcal{P}' = \mathcal{P} = \{x \in \mathbb{R}^n :\ Ax \leq b\}\,.$$

□

Definition 4.5 (Extreme points)**.** *Let $C \subset \mathbb{R}^n$ be a convex set. A point $x \in C$ is an* extreme point *of C if x is not a convex combination of two points in C distinct from x*

$$\nexists\, x_1, x_2 \in C \setminus \{x\}, \lambda \in (0, 1) \text{ s.t. } x = \lambda x_1 + (1 - \lambda)x_2.$$

See Figure 11 for an illustration of this concept.

Theorem 4.6 (Minkowski's Theorem)**.** *If $\mathcal{P} \subset \mathbb{R}^n$ is a polyhedron and bounded, then $\mathcal{P}$ is a polytope, that is, $\mathcal{P}$ has a finite set X of extreme points, and $\mathcal{P} = \operatorname{conv}(X)$.*

Proof. See problem sheet. □

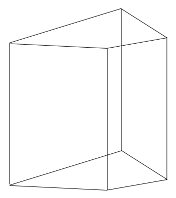


Figure 12: Relaxation of a MIP.

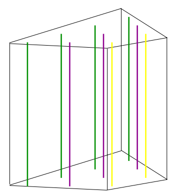


Figure 13: The feasible set is the intersection of the formulation with the integer lattice.

4.3 Alternative Formulations of IP Problems

Let us now consider the mixed integer programming problem

$$\text{(MIP)} \qquad \max_{(x,y)\in\mathbb{R}^{n+p}} c_x^{\mathrm{T}} x + c_y^{\mathrm{T}} y$$
$$\text{s.t.} \quad Ax + By \le b$$
$$x \in \mathbb{Z}^n,$$

where A and B are matrices, and let us denote the set of feasible solutions of (MIP) by

$$\mathscr{F} := \left\{(x,y) \in \mathbb{R}^{n+p} : \, Ax + By \le b, \, x \in \mathbb{Z}^n\right\}.$$

If we drop the integrality constraints $x \in \mathbb{Z}^n$, the set of points that satisfy the remaining constraints is a polyhedron:

$$\mathcal{P} := \left\{(x,y) \in \mathbb{R}^{n+p} : \, Ax + By \le b\right\}.$$

See Figure 12 for an illustration.

Furthermore, we have $\mathscr{F} = \mathcal{P} \cap (\mathbb{Z}^n \times \mathbb{R}^p)$, as depicted in Figure 13

Definition 4.7 (Formulation of a MIP-feasible set)**.** *A polyhedron $\mathcal{P} \subset \mathbb{R}^{n+p}$ is called a* formulation *of a set $\mathscr{F} \subseteq \mathbb{Z}^n \times \mathbb{R}^p$ if*

$$\mathcal{P} \cap \left(\mathbb{Z}^n \times \mathbb{R}^p\right) = \mathscr{F}.$$

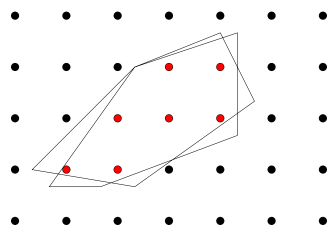


Figure 14: Alternative formulations of the same MIP.

A formulation of Problem (MIP) is any formulation of its feasible set $\mathscr{F}$.

Example 4.8 (Alternative formulations)**.** *Consider the polyhedra*

$$\mathcal{P}_1 = \{x \in \mathbb{R}^4 : 0 \le x \le \mathbf{1},\ 83x_1 + 61x_2 + 49x_3 + 20x_4 \le 100\},$$
$$\mathcal{P}_2 = \{x \in \mathbb{R}^4 : 0 \le x \le \mathbf{1},\ 4x_1 + 3x_2 + 2x_3 + x_4 \le 4\},$$
$$\mathcal{P}_3 = \{x \in \mathbb{R}^4 : 0 \le x \le \mathbf{1},\ 4x_1 + 3x_2 + 2x_3 + x_4 \le 4,\ x_1 + x_2 + x_3 \le 1,\ x_1 + x_4 \le 1\}.$$

Then the IPs

$$(IP_i) \quad \max_{x \in \mathbb{Z}^4} c^{\mathrm{T}} x \quad \text{s.t. } x \in \mathcal{P}_i$$

all describe the same mathematical optimisation problem for $i = 1, 2, 3$*, because all three polyhedra are formulations of the same feasible set*

$$\mathscr{F} = \{(0,0,0,0), (1,0,0,0), (0,1,0,0), (0,0,1,0), (0,0,0,1), (0,1,0,1), (0,0,1,1)\}$$

Algorithmically, however, Formulation 3 is easier to solve than Formulation 2, and the latter is easier to solve than Formulation 1, because $\mathcal{P}_3 \subset \mathcal{P}_2 \subset \mathcal{P}_1$.

Let us verify these claims: If $x \in \mathcal{P}_1 \cap \{0,1\}^4$, then dividing $83x_1 + 61x_2 + 49x_3 + 20x_4 \le 100$ by $20\frac{1}{3}$ yields

$$4.08x_1 + 3x_2 + 2.41x_3 + 0.98x_4 \le 4.92,$$

and since $x_4 \le 1$, adding $0.02x_4$ yields

$$4.08x_1 + 3x_2 + 2.41x_3 + x_4 \le 4.94.$$

Since all $x_i \geq 0$, rounding down the coefficients in the l.h.s. can only render the inequality more satisfied,

$$4x_1 + 3x_2 + 2x_3 + x_4 \leq 4.94.$$

Seen as all x_i are integers, the l.h.s must be integer, hence we can round down the r.h.s.,

$$4x_1 + 3x_2 + 2x_3 + x_4 \leq 4.$$

This shows $\mathcal{P}_1 \cap \{0,1\}^4 \subseteq \mathcal{P}_2 \cap \{0,1\}^4$. Moreover, $x \in \mathcal{P}_2$ implies (multiplying $4x_1 + 3x_2 + 2x_3 + x_4 \leq 4$ by 25),

$$100x_1 + 75x_2 + 50x_3 + 25x_4 \leq 100,$$

and using $x_i \geq 0$,

$$83x_1 + 61x_2 + 49x_3 + 20x_4 \leq 100,$$

whence $x \in \mathcal{P}_1$. Therefore, $\mathcal{P}_2 \subset \mathcal{P}_1$ and $\mathcal{P}_2 \cap \{0,1\}^4 \subseteq \mathcal{P}_1 \cap \{0,1\}^4$, which implies

$$\mathcal{P}_1 \cap \{0,1\}^4 = \mathcal{P}_2 \cap \{0,1\}^4.$$

Next, If $x \in \mathcal{P}_2 \cap \{0,1\}^4$, then $4x_1 + 3x_2 + 2x_3 + x_4 \leq 4$, thus at most one of x_1, x_2, x_3 can be equal to 1, and at most one of x_1, x_4 can be equal to 1. Therefore,

$$\begin{aligned} x_1 + x_2 + x_3 &\leq 1, \\ x_1 + x_4 &\leq 1, \end{aligned}$$

so that $x \in \mathcal{P}_3$. This shows that $\mathcal{P}_2 \cap \{0,1\}^4 \subseteq \mathcal{P}_3 \cap \{0,1\}^4$. It is also immediate that $\mathcal{P}_3 \subset \mathcal{P}_2$, and hence, $\mathcal{P}_3 \cap \{0,1\}^4 \subseteq \mathcal{P}_2 \cap \{0,1\}^4$. Therefore,

$$\mathcal{P}_2 \cap \{0,1\}^4 = \mathcal{P}_3 \cap \{0,1\}^4.$$

Definition 4.9 (Tigher formulation). *If two formulations $\mathcal{P}_1$, $\mathcal{P}_2$ of the feasible set $\mathscr{F}$ of an IP or MIP satisfy $\mathcal{P}_2 \subset \mathcal{P}_1$, we say that $\mathcal{P}_2$ is a* tighter *formulation than $\mathcal{P}_1$.*

Definition 4.10 (Ideal formulation). *If a formulation $\mathcal{P}$ of the feasible set $\mathscr{F}$ satisfies $\mathcal{P} = \mathrm{conv}(\mathscr{F})$, then $\mathcal{P}$ is called an* ideal *formulation. See Figure 15 for an illustration of this concept.*

Example 4.11 (Ideal formulation). *Formulation P_3 of the previous example is an ideal formulation of $\mathscr{F}$.*

Theorem 4.12 (Ideal formulations are solved by LP relaxation). *Let*

$$\begin{aligned} \textit{(IP)} \quad z = \max_x \, & c^{\mathrm{T}} x \\ \text{s.t. } & Ax = b \\ & x \geq 0 \\ & x_j \in \mathbb{Z}, \quad (j = 1, \dots, n) \end{aligned}$$

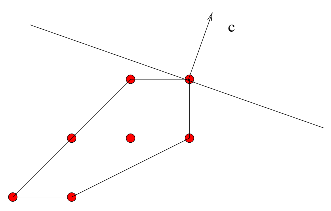


Figure 15: An ideal formulation.

be an IP given with an ideal formulation $\mathcal{P} = \{x : Ax = b,\, x \geq 0\}$, and let x^* be an optimal basic solution (one found by application of the simplex algorithm) of the LP relaxation

$$\textit{(LP)} \quad w = \max_x c^{\mathrm{T}} x \quad \text{s.t. } Ax = b,\quad x \geq 0.$$

Then x^* is also an optimal solution of (IP).

Proof. Since (LP) is a relaxation of (IP), we already know that $z \leq w$, so all we need to establish is that x^* is feasible for (IP), that is, it takes integer values, and optimality follows automatically. Since by assumption $\mathcal{P} = \operatorname{conv}(\mathscr{F})$, each extreme point $\tilde{x}$ of $\mathcal{P}$ is in $\mathscr{F}$, and thus $\tilde{x}$ is integer valued. It therefore suffices to prove that x^* is an extreme point of $\mathcal{P}$.

Since x^* is an optimal basic solution, it is a basic feasible solution, and it thus suffices to prove that any basic feasilbe solution $\tilde{x}$ of $\mathcal{P}$ is an extreme point. Let $\tilde{x}$ be a basic feasible solution with basis B and non-basis N, that is, $\tilde{x}_B = A_B^{-1} b$ and $\tilde{x}_N = 0$. Suppose that

$$\tilde{x} = \lambda x^1 + (1 - \lambda) x^2 \tag{18}$$

for some $x^1, x^2 \in \mathcal{P}$ and $\lambda \in (0, 1)$. We have $x_N^1, x_N^2 \geq 0$ and $0 = \tilde{x}_N = \lambda x_N^1 + (1 - \lambda) x_N^2$, whence $x_N^1 = x_N^2 = 0$. It follows that $b = Ax^i = A_B x_B^i$, so that $x_B^i = A_B^{-1} b = \tilde{x}_B$ for $(i = 1, 2)$. We have shown that $x^1 = x^2 = \tilde{x}$ for all representations of the form (18), which shows that $\tilde{x}$ is an extreme point. $\square$

5 Lecture 5: Total Unimodularity

5.1 Totally Unimodular Matrices

We consider the IP

$$\text{(IP)} \quad \max_x \{c^{\mathrm{T}}x : \, Ax = b, \, x \geq 0, \, x \in \mathbb{Z}^n\}$$

and ask the question whether we have a chance to recognise if

$$\mathcal{P} = \{x \in \mathbb{R}^n : \, Ax = b, \, x \geq 0\}$$

is an ideal formulation, in which case we are in the fortuitous situation that (IP) is solved by its LP relaxation. While this is difficult to decide in general, there exists at least an important family of cases where this is possible:

Definition 5.1 (Totally unimodular matrix)**.** *A matrix $A \in \mathbb{R}^{m \times n}$ is called* totally unimodular *if every square non-singular submatrix of A has determinant ± 1.*

Lemma 5.2 (Extreme point implies basic feasible)**.** *Each extreme point of $\mathcal{P}$ is a basic feasible solution.*

Proof. See problem sheet. □

Theorem 5.3 (Total unimodularity implies integrality I)**.** *If the constraint matrix A of problem (IP) is totally unimodular and b is integer valued, then every extreme point of the formulation $\mathcal{P}$ is integer valued.*

Proof. W.l.o.g., we assume that the rows of A are linearly independent. By the Lemma, every extreme point $\tilde{x}$ is a basic feasible solution, and thus there exists a basis $B = \{j_1, \dots, j_m\}$ and non-basis N such that $\tilde{x}_N = 0$ and

$$A_B \tilde{x}_B = b.$$

The nonbasic components $\tilde{x}_N$ are clearly integer valued, and using Cramer's Rule, we have

$$\tilde{x}_{j_k} = \frac{\det(A_B^k)}{\det(A_B)},$$

where $A_B^k = \left[A_{j_1} \cdots A_{j_{k-1}} \; b \; A_{j_{k+1}} \cdots A_{j_m} \right]$. Since A_B^k has integer components, we have $\det(A_B^k) \in \mathbb{Z}$, and since $\det(A_B) = \pm 1$, it follows that $\tilde{x}_B$ is integer valued. □

5.2 Total Unimodularity Theory

We can easily extend Theorem 5.3 to polyhedra in *inequality constrained form*:

Theorem 5.4 (Total unimodularity implies integrality II)**.** *If A is TU, then for all $b \in \mathbb{Z}^m$, all extreme points of the polyhedron $\mathcal{P}(b) := \{x \in \mathbb{R}^n : Ax \le b,\ x \ge 0\}$ are integer valued.*

Proof. If A is TU, then $\hat{A} = [\,A\ \mathrm{I}\,]$ is TU, so that by Theorem I, the extreme points of

$$\mathcal{P}'(b) := \left\{z \in \mathbb{R}^{n+m} : \hat{A}z = b,\ z \ge 0\right\}$$

are integer valued. Let $\Pi_{\mathbb{R}^n} : z = (x, s) \in \mathbb{R}^{n+m} \mapsto x$ be the projection onto the first n components of the variables z. Then $\mathcal{P}(b) = \Pi_{\mathbb{R}^n}\mathcal{P}'(b)$, and all extreme points of $\mathcal{P}(b)$ are projections of extreme points of $\mathcal{P}'(b)$. Therefore, the extreme points of $\mathcal{P}(b)$ are also integer valued. □

The following is a near-converse result:

Theorem 5.5 (Integrality implies total unimodularity)**.** *If $A \in \mathbb{Z}^{m\times n}$ is such that all extreme points of the polyhedron $\mathcal{P}(b) := \{x \in \mathbb{R}^n : Ax \le b, x \ge 0\}$ are integer valued for all $b \in \mathbb{Z}^m$, then A is TU.*

Proof. Let $\hat{A} = [\,A\ \mathrm{I}\,]$. We will prove in one of the problem sheets that all extreme points of

$$\mathcal{P}'(b) = \left\{z \in \mathbb{R}^{n+m} : \hat{A}z = b,\ z \ge 0\right\}$$

are integer valued. Let $A_{I,J}$ be an arbitrary invertible square submatrix of A, corresponding to row indices I and column indices J. Let

$$B = J \cup \{n + i : i \notin I\},$$

then $\hat{A}_B$ is an invertible $m \times m$ submatrix of $\hat{A}$. If rows I of matrix $\hat{A}_B$ are permuted into the top position by left multiplication with appropriate permutation matrices P_1, P_2, it is of the form

$$P_1\hat{A}_B P_2 = \begin{bmatrix} A_{I,J} & 0 \\ \star & \mathrm{I} \end{bmatrix}.$$

This process is illustrated in Figure 16. Hence $\det(\hat{A}_B) = \pm\det(A_{I,J})$, so that $\det(A_{I,J}) = \pm 1$ if and only if $\det(\hat{A}_B) = \pm 1$.

Let $\mathbf{e}^i$ be the i-th canonical unit vector in $\mathbb{R}^m$, so that

$$\mathbf{1} := \sum_{i=1}^{m} \mathbf{e}^i = \begin{bmatrix} 1 & \dots & 1 \end{bmatrix}^{\mathrm{T}},$$

and let

$$\delta = \left\lceil \max_{k,\ell} \left| \left(\hat{A}_B^{-1}\right)_{k,\ell} \right| \right\rceil.$$

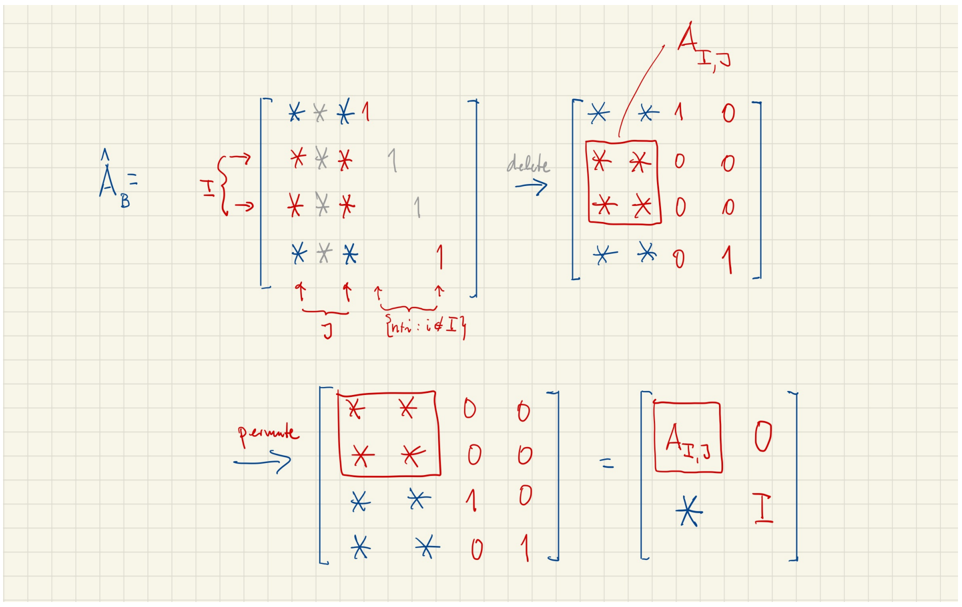


Figure 16: The permutation required in the proof of Theorem 5.5.

Then

$$b^i = \delta \cdot \hat{A}_B \mathbf{1} + \mathbf{e}^i,$$

is an integer vector $\forall i$. The basic solution associated with the basis B and the r.h.s. b^i is

$$x_B = \hat{A}_B^{-1} b^i = \delta \cdot \mathbf{1} + \hat{A}_B^{-1} e_i \geq 0, \quad x_N = 0,$$

so it is basic feasible and hence an extreme point of $\mathcal{P}'$.

$$\begin{aligned}
&\Rightarrow x_B \in \mathbb{Z}^m, \\
&\Rightarrow \left(\hat{A}_B^{-1}\right)_i = \hat{A}_B^{-1}\mathbf{e}^i = x_B - \delta \cdot \mathbf{1} \in \mathbb{Z}^m, \\
&\Rightarrow \hat{A}_B^{-1} = \left[\hat{A}_B^{-1}\mathbf{e}^1 \quad \ldots \quad \hat{A}_B^{-1}\mathbf{e}^m\right] \in \mathbb{Z}^{m\times m}, \\
&\Rightarrow \det\left(\hat{A}_B^{-1}\right) \in \mathbb{Z}, \\
&\Rightarrow \det\left(\hat{A}_B\right), \det\left(\hat{A}_B\right)^{-1} \in \mathbb{Z}, \\
&\Rightarrow \det\left(\hat{A}_B\right) \in \{\pm 1\}.
\end{aligned}$$

□

5.3 Practical Tools to Recognise TU matrices

Verifying that a given matrix is TU seems a task of complexity exponential in the size of the matrix.

There are two categories of simple tools to recognise special cases:

- Rules by which small TU matrices can be assembled into larger ones. By applying the inverse of these rules, we may be able to recognise how to decompose a matrix into smaller parts whose total unimodularity is computationally cheaper to verify.
- Sufficient criteria that can easily be checked may allow us to identify some important families of TU matrices.

The following rules are easy to prove: $A \in \mathbb{R}^{m\times n}$ is TU if and only if any of the following matrices are TU,

i) A^{T},

ii) $\left[\begin{smallmatrix} A & -A \end{smallmatrix}\right]$,

iii) $A \cdot P$, where P is a $n \times n$ permutation matrix

iv) $P \cdot A$, where P is a $m \times m$ permutation matrix,

v) $\left[\begin{smallmatrix} A & J_1 \\ J_2 & 0 \end{smallmatrix}\right]$, with $J_i = P_i \left[\begin{smallmatrix} \mathrm{I} & 0 \\ 0 & 0 \end{smallmatrix}\right] Q_i$, I an identity matrix, 0 a block of zeros, and P_i, Q_i permutation matrices of appropriate size.

Example 5.6 (TU matrix)**.** *The following matrix is TU,*

$$A = \begin{bmatrix} 1 & 0 & 0 & -1 \\ 1 & 1 & -1 & -1 \\ 0 & 0 & 1 & 0 \end{bmatrix}.$$

Indeed, it is trivial to check that $\left[\begin{smallmatrix} 1 & 0 \\ 1 & 1 \end{smallmatrix}\right]$ *is TU. By application of ii),*

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

is TU, and by application of v),

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

is TU. By permuting the last two columns we find that A is TU.

Definition 5.7 (Consecutive ones). *A $0,1$-valued matrix A has the* consecutive ones property *if the rows can be ordered so that the 1s in each column appear consecutively.*

Theorem 5.8 (Consecutive ones implies TU). *If $A \in \{0,1\}^{m\times n}$ has the consecutive ones property, then A is TU.*

Proof. See problem sheet. □

Example 5.9 (Consecutive ones). *The following matrix has the consecutive ones property:*

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \end{bmatrix}$$

Example 5.10 (Workforce planning). *Matrices with the consecutive ones property occur naturally in workforce planning problems:*

- *Workers are assigned to shifts consisting of consecutive time periods in periods $i = 1, \ldots, m$. There are thus at most $\binom{m+1}{2}$ possible shifts $j = 1, \ldots, n$.*
- *Hiring for shift j costs c_j per worker.*
- *In period i at last d_i workers are needed to operate the machinery.*
- *How many workers x_j to hire for each shift so as to minimise the total cost?*

$$\begin{aligned} \min_{x\in\mathbb{R}^n} \; & \sum_{j=1}^{n} c_j x_j \\ \text{s.t.} \; & \sum_{j=1}^{n} a_{ij} x_j \geq d_i, \quad (i = 1, \ldots, m) \\ & x_j \geq 0, \quad (j = 1, \ldots, n) \\ & x_j \in \mathbb{Z}, \quad (j = 1, \ldots, n). \end{aligned}$$

Each column of $A = (a_{ij})$ is of the form $[\,0 \ldots 0\ 1 \ldots 1\ 0 \ldots 0\,]^{\mathrm{T}}$ because all shifts must consist of a set of consecutive time periods. Therefore, the matrix A is TU.

5.4 Another Sufficient Condition for TU

Theorem 5.11 (Sufficient condition). *Let $A = [a_{ij}]$ be a matrix such that*

i) $a_{ij} \in \{+1, -1, 0\}$ for all i, j.

ii) *Each column contains at most two nonzero coefficients,*

$$\sum_{i=1}^{m} |a_{ij}| \leq 2 \quad (j \in [1, n]).$$

iii) *The set M of rows can be partitioned into (M_1, M_2) such that each column j containing* two *nonzero coefficients satisfies*

$$\sum_{i \in M_1} a_{ij} - \sum_{i \in M_2} a_{ij} = 0.$$

Then A is totally unimodular.

Proof. The proof is by contradiction, assuming that A is not TU.

Let B be a smallest submatrix of A such that $\det(B) \notin \{0, +1, -1\}$. Then all columns of B contain exactly two nonzero coefficients, for else there exist permutation matrices P_1, P_2 such that

$$P_1 B P_2 = \begin{bmatrix} \pm 1 & * \\ 0 & C \end{bmatrix},$$

and then $\det(C) = \pm \det(B) \notin \{0, +1, -1\}$, and C is the row permutation of a strict submatrix of B, contradicting the choice of B. ↯

Because of iii), adding the rows of B with indices in M_1 and subtracting the rows with indices in M_2 yields the zero vector, showing that the rows of B are linearly dependent and $\det(B) = 0$, in contradiction to the choice of B. ↯ □

5.5 Application to Graph Problems

Definition 5.12 (Graph). *A graph $G = (V, E)$ consists of a finite set of* vertices *(or* nodes*) V and a finite collection of* edges *$E \subset \{\{v, w\} : v, w \in V\}$ consisting of unordered pairs of vertices, referred to as the* heads *or* endpoints *of the edge.*

If v is a head of e, we say that e and v are incident *to one another.*

An edge $e \in E$ is called a loop *at $v \in V$ if both heads of e equal v.*

The vertex-edge incidence matrix *of G is the matrix $0, \pm 1$-valued matrix*

$$A(G) = \left(A_{v,e}(G)\right)_{v \in V, e \in E},$$

$$A_{v,e}(G) = \begin{cases} 1 & \textit{if } v \textit{ is one of two distinct heads of } e, \\ 2 & \textit{if } e \textit{ is a loop at } v, \\ 0 & \textit{otherwise.} \end{cases}$$

Definition 5.13 (Bipartite graph). *A graph G is* bipartite *if $V = V_1 \dot{\cup} V_2$ is a partition and $E \subset \{\{v, w\} : v \in V_1, w \in V_2\}$.*

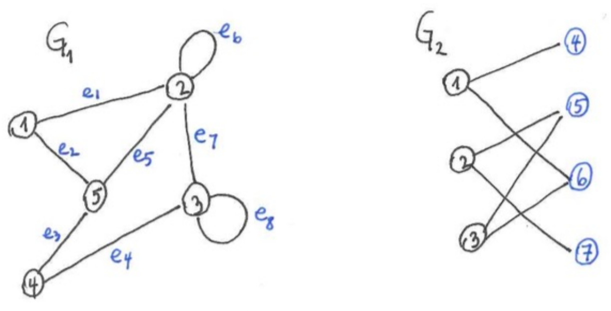


Figure 17: The graphs of Example 5.14.

Example 5.14 (Graph and bipartite graph)**.** *The drawings of Figure 17 give an example of a graph and a bipartite graph with incidence matrices*

$$A(G_1) = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 2 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 2 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \qquad A(G_2) = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

Definition 5.15 (Digraph)**.** *A graph G is a* digraph *(directed graph) if $E \subset \{(v, w) : v, w \in V\}$ consists of* ordered *pairs, giving each edge (or* arc*) a direction from its* tail *v to its* head *w. The vertex-edge incidence matrix is then defined as*

$$A_{v,e}(G) = \begin{cases} 1 & \text{if } v \text{ is the head of } e, \\ -1 & \text{if } v \text{ is the tail of } e, \\ 0 & \text{if } e \text{ is a loop at } v, \\ 0 & \text{otherwise.} \end{cases}$$

Example 5.16 (Digraph)**.** *The graph of Figure 18 is a digraph with incidence matrix*

$$A(G_3) = \begin{bmatrix} 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 1 & -1 \\ 0 & 1 & -1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & -1 & 1 \end{bmatrix}$$

Theorem 5.17 (Incidence matrix of bipartite graph implies TU)**.** *The vertex-edge incidence matrix of any bipartite graph is TU.*

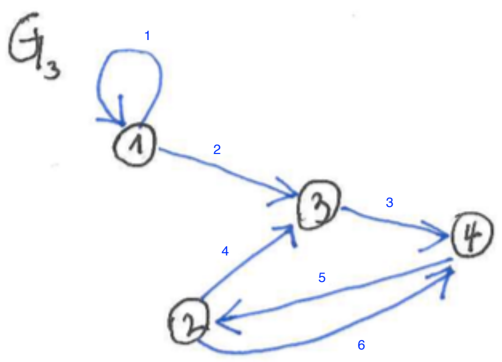


Figure 18: The graph of Example 5.16.

Proof. Each column of $A(G)$ contains exactly two nonzero components, a 1 for some $v \in V_1$, and a 1 for some $w \in V_2$. Therefore, the sufficient criterion of the above theorem applies for the choice $M_1 = V_1$, $M_2 = V_2$. $\square$

Theorem 5.18 (Incidence matrix of digraph implies TU)**.** *The vertex-edge incidence matrix of any digraph is TU.*

Proof. Each column $A_{:,e}(G)$ corresponding to a loop is a zero vector. If e is not a loop, then $A_{:,e}(G)$ contains exactly two nonzero components, a $+1$ for the head, and a -1 for the tail. Therefore, the sufficiency theorem applies with M_1 = M, $M_2 = \emptyset$. $\square$

Example 5.19 (Shortest Path Problem)**.**

- *Given is a digraph $G = (V, E)$ with nonnegative arc lengths c_e for all $e \in E$.*
- *Two nodes $s, t \in V$ are marked.*
- *Find a shortest path from s to t in G.*

For each $e \in E$, let $x_e = 1$ if e lies along the path taken, and $x_e = 0$ otherwise.

For each $v \in V$, let $b_v = 1$ if $v = s$, $b_v = -1$ if $v = t$, and $b_v = 0$ otherwise.

We write $V^+(v), V^{-1}(v)$ for the successor and predecessor nodes of v.

$$\begin{aligned}
\textit{(SP)}\quad z = \min \; & \sum_{(i,j)\in E} c_{ij}x_{ij} \\
\textit{s.t.}\; & \sum_{j\in V^+(s)} x_{sj} - \sum_{j\in V^-(s)} x_{js} = 1 \\
& \sum_{j\in V^+(t)} x_{tj} - \sum_{j\in V^-(t)} x_{jt} = -1 \\
& \sum_{j\in V^+(i)} x_{ij} - \sum_{j\in V^-(i)} x_{ji} = 0 \quad (i \in V \setminus \{s,t\}) \\
& 0 \le x_{ij} \le 1 \quad ((i,j)\in E) \\
& x \in \mathbb{Z}^{|E|}.
\end{aligned}$$

In matrix form, we now have

$$\begin{aligned}
\textit{(SP)}\quad \min\; & \sum_{e\in E} c_e x_e \\
\textit{s.t.}\; & A(G)x = b, \\
& 0 \le x_e \le 1, \quad (e \in E), \\
& x_e \in \mathbb{Z}, \quad (e \in E).
\end{aligned}$$

The constraint matrix of (SP) (reformulated in inequality constrained form) is $[A^{\mathrm{T}}, -A^{\mathrm{T}}, \mathrm{I}]^{\mathrm{T}}$, in which $A = A(G)$ is the vertex-edge incidence matrix of a digraph, hence the model is TU and may be solved via LP relaxation.

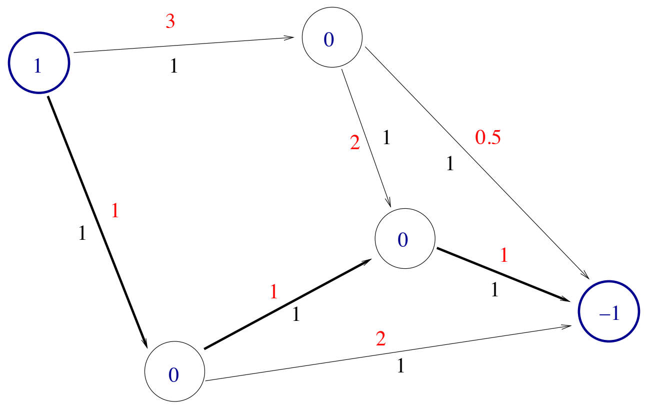


Figure 19: Shortest path problem as a network flow problem.

Note: (SP) has an interpretation as an s-t flow problem with capacities 1 on each edge and integrality constraints on the x_e, with flow conservation constraints at each vertex. See Figure 19 for an illustration.

Example 5.20 (Assignment Problem)**.** *The problem lives in a bipartite graph* $G =$

(V_1, V_2, E) *with* $V_1 = \{i_1, \dots, i_n\}$ *workers,* $V_2 = \{j_1, \dots, j_n\}$ *jobs,* $E = V_1 \times V_2$.

$$\begin{aligned}
\min_{x \in \mathbb{R}^{n \times n}} \quad & \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij} \\
\textit{s.t.} \quad & \sum_{j=1}^{n} x_{ij} = 1 \quad \textit{for } i = 1, \dots, n, \\
& \sum_{i=1}^{n} x_{ij} = 1 \quad \textit{for } j = 1, \dots, n, \\
& x_{ij} \in \{0, 1\} \quad \textit{for } i, j = 1, \dots, n.
\end{aligned}$$

Reformulating the problem in terms of the vertex-edge incidence matrix $A(G)$,

$$\begin{aligned}
\min_{x \in \mathbb{R}^{n \times n}} \quad & \sum_{e \in E} c_e x_e \\
\textit{s.t.} \quad & A(G) x = \mathbf{1}, \\
& 0 \leq x_e \leq 1, \quad (e \in E), \\
& x_e \in \mathbb{Z}, \quad (e \in E),
\end{aligned}$$

we recognise the problem as totally unimodular.

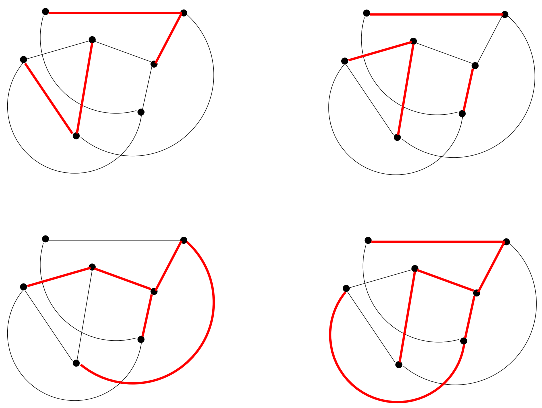
6 Lecture 6: Submodularity

6.1 The Maximum Weight Forest Problem

Definition 6.1 (Subgraph)**.** *A* subgraph *of a graph $G = (V, E)$ is a graph $G' = (V', E')$ such that $V' \subseteq V$ and $E' \subseteq E \cap \{\{u, v\} : u, v \in V'\}$.*

Definition 6.2 (Forest)**.** *A* forest *in a graph $G = (V, E)$ is a subgraph $G' = (V', E')$ that contains no cycles, that is, every pair of nodes are connected via at most one path in G'.*

Example 6.3 (Forests)**.** *The red edges in the following graphs form forests.*



Let $G = (V, E, c)$ be a graph with $n = |V|$ vertices and $m = |E|$ edges, endowed with a set of edge weights $c_e \in \mathbb{R}$ for every $e \in E$.

Definition 6.4 (MWF)**.** *The Maximum Weight Forest Problem finds a forest of maximal total weight in G,*

$$\textit{(MWF)} \quad \max\left\{\sum_{e\in E'} c_e : E' \subseteq E \text{ s.t. } G' = (V, E') \textit{ is a forest}\right\}.$$

We will later reformulate this combinatorial optimisation problem as an IP and see that the following $O(n + m\log m)$-time algorithm solves (MWF) to optimality.

Algorithm 6.5 (Greedy MWF)**.**

```
// initialisation
order edges $c_1 \geq \cdots \geq c_m$ by nonincreasing weights ;
set $F^0 := \emptyset$;
initialise function cycles;
// body
for $i = 1, \dots, m$ do
    if $c_i \leq 0$ then
        return "$F^{i-1}$ optimal";
    else if $F^{i-1} \cup \{e_i\}$ contains no cycle then
        $F^i := F^{i-1} \cup \{e_i\}$;
    else
        $F^i := F^{i-1}$;
    end
    if $|F^i| = n - 1$ or $i = m$ then
        return "$F^i$ optimal";
    end
end
```

Notes:

- $n - 1$ is the maximal possible cardinality of a forest.
- The second step in the body requires checking if adding the edge e_t creates a cycle, which can be checked by calling Function 6.6 below.

Function 6.6 (Cycles)**.**

```
// initialisation
create lists (connected components) $L_1 = \{v_1\}, \dots, L_n = \{v_n\}$;
create pointers $v \mapsto L(v)$ to lists;
// input
$e = \{v, w\}$;
// body
if $L(v) = L(w)$ then
    return "creates cycle"
else
    merge $L(v)$ and $L(w)$;
    return "creates no cycle";
end
```

6.2 Submodular functions

Definition 6.7 (Submodularity)**.** *Let $\mathscr{P} := \{A : A \subseteq \{1, \dots, n\}\}$ for some $n \in \mathbb{N}$. A function $f : \mathscr{P} \to \mathbb{R}$ is called*

i) submodular *if*

$$f(A) + f(B) \geq f(A \cap B) + f(A \cup B), \qquad \forall\, A, B \in \mathscr{P},$$

ii) non-decreasing *if*

$$f(A) \leq f(B) \qquad \forall\, A \subseteq B \in \mathscr{P}.$$

Example 6.8 (Submodularity)**.**

- $f : A \mapsto |A|$ *is submodular non-decreasing.*
- *Let* $\{v_1, \dots, v_n\} \subset V$ *be vectors in a vector space* V*. Then*

  $$f : A \mapsto \operatorname{rank}(\{v_i : i \in A\})$$

  is submodular non-decreasing, where $\operatorname{rank}(\{v_i : i \in A\}) := \dim(\operatorname{span}\{v_i : i \in A\})$*.*
- *Let* $\{v_1, \dots, v_n\}$ *be a list of training samples (images, audio files, texts, ...),* $G_i \in \mathscr{P}$ $(i = 1, \dots, k)$ *a list of groups of samples* $\{v_i : i \in G_i\}$*, and* $h : \mathbb{R}_+ \to \mathbb{R}_+$ *a concave non-decreasing function. Then*

  $$f : A \mapsto \sum_{i=1}^{k} h\left(|G_i \cap A|\right)$$

  is a submodular non-decreasing function (structured diversity function used in machine learning, image segmentation, etc.).

6.3 Submodular Optimisation

Definition 6.9 (Submodular Polyhedron)**.** *Let* f *be a non-decreasing submodular function for which* $f(\emptyset) = 0$*. The* submodular polyhedron *associated with* f *is defined by*

$$P(f) = \Big\{x \in \mathbb{R}^n : x \geq 0,\ \sum_{j \in S} x_j \leq f(S),\ \forall S \in \mathscr{P} \setminus \{\emptyset\}\Big\}.$$

Given a vector $c \in \mathbb{R}^n$*, the* submodular optimisation problem *associated with* f *and* c *is given by*

$$\text{(SOP)} \quad \max\{c^{\mathrm{T}} x : x \in P(f)\}.$$

Note that (SOP) is an LP with $2^n - 1$ constraints. Therefore, even for moderately sized n the constraint matrix of (SOP) could not be held explicitly in the memory of a computer, and the simplex method could not be applied.

Suprisingly, an efficient algorithm for solving (SOP) is nonetheless available:

Algorithm 6.10 (Greedy SOP)**.**

```
// initialisation
re-indexation s.t. $c_1 \geq \cdots \geq c_r > 0 \geq c_{r+1} \geq \cdots \geq c_n$, where $r = n$ if $c_n > 0$;
set $S^0 := \emptyset$;
// body
for $i = 1, \dots, r$ do
    set $S^i = \{1, \dots, i\}$;
    set $x_i = f(S^i) - f(S^{i-1})$;
end
for $j > r$ do
    set $x_j = 0$;
end
```

Lemma 6.11 (Law of diminishing returns)**.** *Let $C \subset D \in \mathscr{P}$ such that $D \setminus C = \{d\}$ is a singleton. Then for all $T \in \mathscr{P}$ with $d \in T$, we have*

$$f(D) - f(C) \leq f(D \cap T) - f(C \cap T).$$

Proof. Applying the submodularity inequality to $A = C$ and $B = D \cap T$,

$$\begin{aligned} f(C) + f(D \cap T) &= f(A) + f(B) \\ &\geq f(A \cap B) + f(A \cup B) \\ &= f(C \cap T) + f(D), \quad \text{(since } d \in T\text{)}. \end{aligned}$$

which proves the claim. □

Theorem 6.12 (Greedy solution of SOP)**.** *The greedy algorithm terminates in $\mathscr{O}\big(n(w_f + \log n)\big)$ time, where w_f is the amount of work required per function evaluation of f, and its output x is an optimal solution for (SOP).*

Proof. The complexity claim is immediate, as steps 2 and 3 take at most $n \cdot w_f$ time, and step 1 takes $\mathscr{O}(n \log n)$ time.

Since f is nondecreasing, $x_i = f(S^i) - f(S^{i-1}) \geq 0$ $(i = 1, \dots, r)$, and since $x_i = 0$ $(i = r+1, \dots, n)$, the nonnegativity constraints $x \geq 0$ are satisfied. Further, for all $T \in \mathscr{P}$,

$$\begin{aligned} \sum_{j \in T} x_j &= \sum_{j \in T \cap S^r} \big(f(S^j) - f(S^{j-1})\big) \quad \text{(by construction of } x\text{)} \\ &\leq \sum_{j \in T \cap S^r} \big(f(S^j \cap T) - f(S^{j-1} \cap T)\big) \quad \text{(law of diminishing returns)} \\ &\leq \sum_{j \in S^r} \big(f(S^j \cap T) - f(S^{j-1} \cap T)\big) \quad (f \text{ is non-decreasing}) \\ &= f(S^r \cap T) - f(\emptyset) \leq f(T). \quad \text{(telescoping sum)} \end{aligned}$$

Therefore, x is (SOP)-feasible with objective value $z^* = \sum_{i=1}^r c_i \big(f(S^i) - f(S^{i-1})\big)$.

To show optimality, we construct a dual feasible solution with dual objective value z^*. The dual of reads

$$\begin{aligned}
\text{(D)} \quad \min & \sum_{S \in \mathscr{P}} f(S) \cdot y_S \\
\text{s.t.} & \sum_{S : j \in S} y_S \geq c_j \quad (j = 1, \dots, n) \\
& y_S \geq 0 \quad \forall\, S \in \mathscr{P}.
\end{aligned}$$

For $i = 1, \dots, r-1$ set $y_{S^i} = c_i - c_{i+1}$. Set $y_{S^r} = c_r$, and set $y_S = 0$ for all other $S \in \mathscr{P}$. Then y is (D)-feasible, since $y_S \geq 0$ for all $S \in \mathscr{P}$, and furthermore for $j \leq r$,

$$\sum_{S : j \in S} y_S = \sum_{i=j}^{r} y_{S^i} = \sum_{i=j}^{r-1} (c_i - c_{i+1}) + c_r = c_j,$$

whereas for $j > r$,

$$\sum_{S : j \in S} y_S = 0 \geq c_j.$$

The dual objective value at y is

$$\sum_{i=1}^{r} f(S^i) y_{S^i} = \sum_{i=1}^{r-1} f(S^i)(c_i - c_{i+1}) + f(S^r) c_r = \sum_{i=1}^{r} c_i \left(f(S^i) - f(S^{i-1}) \right) = z^*,$$

as claimed. □

6.4 Submodular Rank Functions

Definition 6.13 (Submodular rank function). *$f : \mathscr{P} \to \mathbb{R}$ is called a* submodular rank function *if the following are satisfied,*

i) *f is non-decreasing submodular*

ii) $f(\emptyset) = 0$,

iii) $f(S \cup \{j\}) - f(S) \in \{0, 1\}$*, for all $S \in \mathscr{P}$ and $j \notin S$.*

Example 6.14 (Submodular rank function). *The following are examples of submodular rank functions:*

- *Let $\{v_1, \dots, v_n\} \subset V$ be vectors in a vector space V. Then*

$$f : A \mapsto \operatorname{rank}(\{v_i : i \in A\})$$

is a submodular rank function.

- *Let $G = (V, E)$ be an undirected graph, and let*

$$f : \mathscr{P}(E) \to \mathbb{R},$$
$$A \mapsto \max\big\{|F| : F \subseteq A,\ (V, F) \text{ is a forest}\big\}.$$

  Then f is a submodular rank function.

Proposition 6.15 (Properties of submodular rank functions)**.** *If f is a submodular rank function, then*

i) *$f(A) \leq |A|$ for all $A \in \mathscr{P}$,*

ii) *If $f(A) = |A|$ then $f(B) = |B|$ for all $B \subset A$.*

iii) *Let x^A be the incidence vector of $A \in \mathscr{P}$, defined by $x^A_j = 1$ if $j \in A$ and $x^A_j = 0$ otherwise. Then $x^A \in P(f)$ if and only if $f(A) = |A|$.*

Proof. i) Let $A = \{a_1, \dots, a_\ell\}$ and $A_k := \{a_1, \dots, a_k\}$, $(k = 1, \dots, \ell)$. Then

$$f(A) = f(A_\ell) \leq f(A_{\ell-1})+1 \leq (f(A_{\ell-2})+1)+1 \leq \dots ((\dots(f(\emptyset)+1)\dots))+1 = \ell = |A|. \tag{19}$$

ii) W.l.o.g. the elements of A are indexed so that $B = A_k$ for some $k < \ell$. If $f(A) = |A|$, then all inequalities in (19) must hold as equalities, and then $f(A_k) = |A_k|$ for all k. In particular, $f(B) = |B|$.

iii) If $f(A) < |A|$ then $\sum_{j \in A} x^A_j = |A| > f(A)$, and hence, $x^A \notin P(f)$. Conversely, if $f(A) = |A|$, then for all $S \in \mathscr{P}(N)$,

$$\sum_{j \in S} x^A_j = |A \cap S| \overset{ii)}{=} f(A \cap S) \leq f(S).$$

Hence, $x^A \in P(f)$, as claimed. □

7 Lecture 7: Matroids

7.1 Submodular Rank Functions Continued

Lemma 7.1 (Closure). *Let $f : \mathscr{P} \to \mathbb{R}$ be a submodular rank function. If $A, B \in \mathscr{P}$ are such that $f(A \cup \{e\}) = f(A)$ for all $e \in B \setminus A$, then*

$$f(A \cup B) = f(A).$$

Proof. See Problem Sheet 2. □

Definition 7.2 (Independent set). *Let $N := \{1, \dots, n\}$ be a finite set and $f : \mathscr{P}(N) :\to \mathbb{N}$ a submodular rank function. A set $A \in \mathscr{P}(N)$ is called* independent *under f if $f(A) = |A|$.*

Proposition 7.3 (Properties of independent sets). *The set $\mathcal{I} = \{A \in \mathscr{P} : |A| = f(A)\}$ of independent sets satisfies*

i1) $\emptyset \in \mathcal{I}$,

i2) if $A \subset B$ and $B \in \mathcal{I}(M)$, then $A \in \mathcal{I}(M)$,

i3) if $A, B \in \mathcal{I}$ and $|B| > |A|$, then $\exists\, e \in B \setminus A$ such that $A \cup \{e\} \in \mathcal{I}$.

Proof. Property i1) follows from Part i) of Proposition (Properties of submodular rank functions), while i2) follows from Part ii). If i3) doesn't hold, then by Lemma (Closure), $f(A \cup B) = f(A)$, and then

$$|B| > |A| = f(A) = f(A \cup B) \geq f(B) = |B|. \lightning$$

□

7.2 Matroids

Definition 7.4 (Matroid). *A* matroid *$M = \big(E(M), \mathcal{I}(M)\big)$ consists of*

- *a finite set $E(M)$ (the* ground set *of M),*
- *a subset $\mathcal{I}(M)$ of the power set $\mathscr{P}\big(E(M)\big) = \{A : A \subseteq E(M)\}$ that satisfies properties*

  i1) $\emptyset \in \mathcal{I}$,

  i2) if $A \subset B$ and $B \in \mathcal{I}(M)$, then $A \in \mathcal{I}(M)$,

  i3) if $A, B \in \mathcal{I}$ and $|B| > |A|$, then $\exists e \in B \setminus A$ such that $A \cup \{e\} \in \mathcal{I}$.

The rank function *$r_M : \mathscr{P}\big(E(M)\big) \to \mathbb{R}$ of M is defined by*

$$A \mapsto \max\{|X| : X \subseteq A,\ X \in \mathcal{I}(M)\}.$$

Notes:

- W.l.o.g., $E(M) = \{1, \dots, n\}$.
- Axioms i1) – i3) generalise the properties of sets of linearly independent vectors.
- Matroids generalise aspects of both matrices and graphs.

Example 7.5 (Linear matroid)**.** *Let $A = [\, v_1 \cdots v_n \,]$ be a matrix over a field F (e.g. $F = \mathbb{R}$). Let*

$$E(M) = \{1, \dots, n\},$$
$$\mathcal{I}(M) := \{X \subseteq \{1, \dots, n\} : \{v_i : i \in X\} \textit{ linearly independent}\}.$$

Then $M = \big(E(M), \mathcal{I}(M)\big)$ is the linear matroid *of A, and we say that A is a* representation *of M over F.*

Example 7.6 (Graphic matroid)**.** *Let $G = (V, E)$ be a graph, and let $E(M) = E$ and $\mathcal{I}(M)$ be the set of forests in G. Then $M = \big(E(M), \mathcal{I}(M)\big)$ is the* graphic matroid *of G.*

Theorem 7.7 (Matroid induced by submodular rank function)**.** *Let $E := \{1, \dots, n\}$, $f : \mathscr{P}(E) \to \mathbb{R}$ a submodular rank function and $\mathcal{I} := \{A \in \mathscr{P}(E) : f(A) = |A|\}$. Then $M = (E, \mathcal{I})$ is a matroid with $E(M) = E$, $\mathcal{I}(M) = \mathcal{I}$ and rank function $r_M = f$.*

Proof. Proposition (Properties of independent sets) shows that $M = (E, \mathcal{I})$ is a matroid. Furthermore, the rank function associated with M is

$$r_M : A \mapsto \max\{|X| : X \subseteq A, X \in \mathcal{I}\}. \tag{20}$$

Let X_A be a maximising independent subset of A in (20). Then $f(X_A) = |X_A|$, and for all $e \in A \setminus X_A$, we have $X_A \cup \{e\} \notin \mathcal{I}$ and thus,

$$f(X_A) \leq f(X_A \cup \{e\}) < |X_A \cup \{e\}| = |X_A| + 1 = f(X_A) + 1,$$

which implies $f(X_A \cup \{e\}) = f(X_A)$. By Lemma (Closure), this implies

$$f(A) = f(X_A \cup A) = f(X_A) = |X_A| = r_M(A),$$

as claimed. □

Theorem 7.8 (Submodular rank function induced by matroid)**.** *Let $M = (E(M), \mathcal{I}(M))$ be a matroid. Then r_M is a submodular rank function with induced matroid M.*

Proof. See Problem Sheet 2. □

7.3 The MISP

Definition 7.9 (MISP)**.** *Let $M = \big(E, \mathcal{I}\big)$ be a matroid, and let each element e of E be associated with a weight c_e. The* maximum weight independent set problem *(MISP) associated with M and c is given by*

$$\textit{(MISP)} \quad \max\left\{\sum_{j\in A} c_j \,:\, A\in\mathcal{I}\right\}.$$

Theorem 7.10 (Greedy solution of MISP)**.** *The greedy algorithm applied to $f = r_M$ and weights c solves (MISP) to optimality.*

Proof. By Theorem 7.8, r_M is a submodular rank function, and by Theorem (Greedy solution of SOP), the greedy algorithm solves the submodular optimisation problem associated with r_M and c to optimality,

$$\text{(SOP)} \quad \max\left\{\sum_{j\in N} c_j x_j \,:\, x\in P(r_M)\right\}.$$

Since r_M is a rank function, the output x^* of the greedy algorithm applied to (SOP) satisfies $x^*_j \in \{0,1\}$ for all j, by way of how the greedy algorithm is constructed: $x_i = r_M(S^i) - r_M(S^{i-1})$, $(i = 1,\dots,r)$.

Furthermore, it follows from Proposition 6.15 that

$$\begin{aligned}\text{(MISP)} \quad \max\left\{\sum_{j\in A} c_j \,:\, A\in\mathcal{I}\right\} &= \max\left\{\sum_{j\in N} x^A_j c_j \,:\, A\in\mathcal{I}\right\}\\ &= \max\left\{\sum_{j\in N} x_j c_j \,:\, x\in\{0,1\}^n,\, x\in P(r_M)\right\}.\end{aligned}$$

This shows that (SOP) is the LP relaxation of (MISP). Since the optimal solution x^* of (SOP) is (MISP)-feasible, it follows by relaxation that x^* is also (MISP)-optimal. ☐

Example 7.11 (Greedy solution of MWF)**.** *Let $G = (V, E)$ be an undirected graph, and let*

$$\begin{aligned} r_M : \mathscr{P}(E) &\to \mathbb{R},\\ A &\mapsto \max\big\{|F| : F\subseteq A,\ (V,F) \text{ is a forest}\big\}.\end{aligned}$$

Then r_M is a submodular rank function. The associated set of independent sets $\mathcal{I}$ is the set of forests in G, and the associated maximum weight independent set problem becomes the maximum weight forest problem for G.

Step 2.ii) of the greedy algorithm for (SOP) reads

$$x_i = r_M(S^i) - r_M(S^{i-1}) = \begin{cases} 1 & \text{if } F^{i-1} \cup \{e_i\} \text{ is a forest}, \\ 0 & \text{if } F^{i-1} \cup \{e_i\} \text{ contains a cycle}, \end{cases}$$

where F^i denotes the forests built in the greedy algorithm for (MWF), which satisfy $F^i \in \arg\max\{|F| : F \subseteq S^i, (V, F) \text{ is a forest}\}$.

Furthermore, for $j > r$, the greedy algorithm for (SOP) sets $x_j = 0$, which coincides with the effect of the stopping criterion of step 2.i) of the greedy algorithm for (MWF). Stopping criterion 2.iii) of the latter applies because if the main loop were continued, step 2.ii) would always result in the detection of a cycle, and hence $F^i = F^{i-1}$, and hence $x_i = 0$. Therefore, the greedy algorithm for (SOP) reduces exactly to the greedy algorithm for (MWF) in this case.

Definition 7.12 (Matroid polytope). *Let $M = (E(M), \mathcal{I}(M))$ be a matroid. The* matroid polytope *associated with M is defined as*

$$\mathcal{P}_{\mathcal{I}(M)} := \operatorname{conv}\left(x^A \in \{0,1\}^n : A \in \mathcal{I}(M)\right).$$

Corollary 7.12.1 (Matroid polytope and submodular polyhedron). *Let f be a submodular rank function. Then the submodular polyhedron $P(f)$ equals the matroid polytope of the matroid associated with f.*

Proof. By Proposition (Properties of submodular rank functions) iii), $x \in \{0,1\}^n \in P(f)$ if and only if $x = x^A$ for some $A \in \mathcal{I}$, and since $P(f)$ is convex, this shows that

$$\mathcal{P}_{\mathcal{I}(M)} \subseteq P(f).$$

Furthermore, Theorem (Greedy solution of MISP) shows that every linear objective function is maximised over $P(f)$ by a point in $\mathcal{P}_{\mathcal{I}(M)}$. Hence, all extreme points of $P(f)$ are in $\mathcal{P}_{\mathcal{I}(M)}$, and

$$P(f) \subseteq \mathcal{P}_{\mathcal{I}(M)}.$$

□

7.4 Lovasz Extension

Lemma 7.13 (Ordered decomposition). *Every nonzero vector $x \in [0,1]^n$ has a unique decomposition $x = \sum_{j=1}^m \lambda_j x^j$ such that $m \leq n$, $\lambda_j > 0$, $(j = 1, \dots, m)$, $x^j \in \{0,1\}^n$, $(j = 1, \dots, m)$, and*

$$x^1 \geq x^2 \geq \cdots \geq x^m \neq 0.$$

Proof. The proof is by a constructive algorithm. We make the stronger claim that $m \le k(x) := |S(x)|$, where $S(x) := \{i : x_i > 0\}|$ is the support set of x, and that $S(x^m) \subsetneq S(x^{m-1}) \subsetneq \cdots \subsetneq S(x^1)$.

Let $i^* \in \arg\min\{x_i : i \in S(x)\}$. Set $\lambda_1 = x_{i^*}$ and $x^1 = \sum_{i \in S(x)} e^i$. Then $S(x^1) = S(x)$, and we have $y := x - \lambda_1 x^1 \in [0,1]^n$ with $S(y) \subsetneq S(x)$.

If $k(y) = 0$, we are done. Otherwise, $0 < k(y) < k(x)$, and by induction there exists a unique decomposition $y = \sum_{j=2}^m \lambda_j x^j$ with $m - 1 \le k(y) \le k(x) - 1$ and such that

$$S(x^m) \subsetneq S(x^{m-1}) \subsetneq \cdots \subsetneq S(x^2) = S(y) \subsetneq S(x^1),$$

which proves that $x^1 \ge x^2, \cdots \ge x^m$ and $x = \sum_{j=1}^m \lambda_j x^j$.

By definition of i^*, we must have $\lambda_1 = x_{i^*}$ for every decomposition $x = \sum_{j=1}^m \lambda_j x^j$, so that $\lambda_1 x^1$ is unique. Then y is unique, and by induction, the part $\sum_{j=2}^m \lambda_i x^i$ is unique. □

Definition 7.14 (Lovasz Extension)**.** *Let $f : \mathscr{P}(\{1,\dots,n\}) \to \mathbb{R}$ be a submodular non-decreasing function with $f(\emptyset) = 0$. The* Lovasz Extension *of f is the function*

$$f_L : [0,1]^n \to \mathbb{R},$$
$$x \mapsto \sum_{j=1}^m \lambda_j f(S(x^j)),$$

where $x = \sum_{j=1}^m \lambda_j x^j$ is the unique decomposition of Lemma (Ordered decomposition) and $S(x^j)$ is the support of x^j.

Theorem 7.15 (Convexity and integer minima of Lovasz Extension)**.** *The function f_L is convex, piecewise linear, and attains its minimum over $[0,1]^n$ at a point in $\{0,1\}^n$.*

Remark 7.16 (Minimisation of submodular functions)**.** *As a consequence, submodular functions may be minimised using an algorithm that can minimise convex piecewise linear functions, such as the subgradient algorithm (see Lecture 13). But we note that f_L consists of exponentially many linear pieces!*

Proof. (Proof of Theorem 7.15) Let $x \in [0,1]^n$. We claim that $f_L(x) = w(x)$, where

$$w(x) = \max_z \left\{ \sum_{i=1}^n x_i z_i : z \in P(f) \right\}. \tag{21}$$

W.l.o.g. (after re-indexing), $x_1 \ge x_2 \ge \cdots \ge x_n$. Let $S_0 = \emptyset$ and $S_j = \{1,\dots,j\}$ for $j = 1,\dots,n$.

The proof of Theorem 6.12 shows that

$$w(x) = \sum_{i=1}^n x_i \left(f(T_j) - f(T_{j-1})\right) = \sum_{i=1}^n (x_i - x_{i+1}) f(T_j),$$

where $x_{n+1} := 0$ and $T_j = \{1, \ldots, j\}$. Writing $\lambda_j := x_j - x_{j+1}$, we get the decomposition

$$x = \sum_{j\colon \lambda_j > 0} \lambda_j x^{T_j},$$

where x^{T_j} is the indicator vector of the set T_j. By uniqueness this must be the unique decomposition of x according to Lemma (Ordered decomposition). Therefore, as claimed, we have

$$f_L(x) = \sum_{j\colon \lambda_j > 0} \lambda_j f(T_j) = w(x).$$

The proof of Theorem 7.10 applied to the SOP (21) with objective weights x shows that

$$\arg\max_z \left\{ \sum_{i=1}^n x_i z_i : \, z \in P(f) \right\} \in \{0,1\}^n,$$

so that by Proposition (Properties of rank functions),

$$f_L(x) = \max\{z^{\mathrm{T}} x : \, z = x^A, \, A \in \mathcal{I}\}$$

is a pointwise maximum of linear functions. The pointwise maximum of a finite set of linear functions is convex and piecewise linear.

Finally, suppose f_L is not minimised by an integer point, and let $x^* = \arg\min_{x \in [0,1]^n} f_L(x)$. Since $f_L(0) = f(\emptyset) = 0$, we have $f_L(x^*) < 0$. Let $x^* = \sum_{j=1}^m \lambda_j x^j$ be the decomposition of x^* according to Lemma (Ordered decomposition). Then $\lambda_j > 0$ and $f(x^j) \geq 0$, so that

$$0 > f_L(x^*) = \sum_{j=1}^m \lambda_j f(x^j) \geq 0. \quad \lightning$$

$\square$

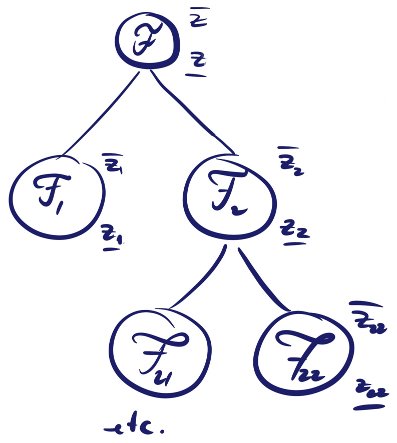


Figure 20: Branch and bound splitting.

8 Lecture 8: Branch and Bound

8.1 Branch-and-Bound: A Divide and Conquer Strategy

LP based Branch-and-Bound is a divide-and-conquer strategy to solve any IP via LP relaxation. We write

$$(\mathscr{F}) \quad z = \max\{c^{\mathrm{T}}x : x \in \mathscr{P}, x \in \mathbb{Z}^n\}$$

for some formulation (polyhedron) $\mathscr{P}$ and feasible set $\mathscr{F} = \mathscr{P} \cap \mathbb{Z}^n$ and with LP relaxation

$$(\mathscr{P}) \quad \overline{z} = \max\{c^{\mathrm{T}}x : x \in \mathscr{P}\},$$

yielding a dual bound $\overline{z}$ that we track. We also keep tracking a primal bound $\underline{z}$ (may be $-\infty$). The main idea of the method is a hierarchical splitting of $\mathscr{F}$ into smaller subsets.

Specifically, we

1. solve $(\mathscr{P})$ to find $\overline{z}$, possibly find $\underline{z}$,
2. split the feasible set so that $\mathscr{F}_1 \cup \mathscr{F}_2 = \mathscr{F}$, but $\mathscr{P}_1 \cup \mathscr{P}_2 \subsetneq \mathscr{P}$, to guarantee that $\overline{z}_1, \overline{z}_2 < \overline{z}$,
3. if $\underline{z}_2 \geq \overline{z}_1$, no need to split $\mathscr{F}_1$ (pruning),
4. repeat with a subproblem stored at another node.

8.2 Introductory Example

Example 8.1. *For illustration, we will solve the IP instance*

$$
\begin{aligned}
(\mathscr{F}) \quad z = \max\, & 4x_1 - x_2 \\
\textit{s.t. } & 7x_1 - 2x_2 \le 14 \\
& x_2 \le 3 \\
& 2x_1 - 2x_2 \le 3 \\
& x \in \mathbb{Z}^2_+ .
\end{aligned}
$$

Step 1: Bounding $(\mathscr{F})$. To obtain a dual bound, we solve the LP relaxation

$$
\begin{aligned}
(\mathscr{P}) \quad \overline{x} = \arg\max\, & 4x_1 - x_2 \\
\text{s.t. } & 7x_1 - 2x_2 \le 14 \\
& x_2 \le 3 \\
& 2x_1 - 2x_2 \le 3 \\
& x \ge 0
\end{aligned}
$$

of $(\mathscr{F})$. Introducing slack variables x_3, x_4, x_5 and applying the simplex algorithm, we obtain the optimal tableau

1	0	1/7	2/7	0	20/7
0	1	0	1	0	3
0	0	-2/7	10/7	1	23/7
0	0	-4/7	-1/7	0	-59/7

We have $\overline{z} = 59/7$, $\underline{z} = -\infty$.

To obtain a primal bound, we could try finding a feasible solution $\tilde{x}$ of (IP) via a heuristic and set $\underline{z} = 4\tilde{x}_1 - \tilde{x}_2$. We would also keep $\tilde{x}$ in memory as *incumbent*. In this case, we did not produce a primal bound via a heuristic and can use $\underline{z} = -\infty$ as a valid lower bound.

Key Idea 8.2 (Tightening Bounds)**.** *So far we know that*

$$\underline{z} \le z \le \overline{z}.$$

In subsequent iterations we will produce improved primal bounds $\underline{z} \le \underline{z}_+$ and improved dual bounds $\overline{z}_+ \le \overline{z}$ such that

$$\underline{z}_+ \le z \le \overline{z}_+$$

sandwiches the optimal objective value z in a narrower interval.

Each time the primal bound $\underline{z}$ increases, a new best feasible solution of $(\mathscr{F})$ has been found, and we update the incumbent.

Should we ever encounter a situation where $\underline{z} = \overline{z}$, the incumbent must be an optimal solution of $(\mathscr{F})$, and we can stop.

Often, we run the algorithm on a fixed time budget and stop early. The bounds then give an approximation guarantee, as the objective value $\underline{z}$ of the incumbent is at most $\overline{z} - \underline{z}$ away from optimal or, assuming positive objective values, the incumbent is within a factor of $\underline{z}/\overline{z}$ of optimal.

Step 2: Branching $(\mathscr{F})$**.**

- $\underline{z} < \overline{z} \Rightarrow (\mathscr{F})$ not soved to optimality.

- Fractional branching: $\overline{x}_1 = 20/7$ is not integer.

$$\mathscr{P}_1 = \mathscr{P} \cap \{x : x_1 \leq \lfloor \frac{20}{7} \rfloor\}, \qquad \mathscr{P}_2 = \mathscr{P} \cap \{x : x_1 \geq \lceil \frac{20}{7} \rceil\}.$$

More generally, we can pick any index j such that $\overline{x}_j \notin \mathbb{Z}$ and set

$$\mathscr{F}_1 = \mathscr{F} \cap \{x : x_j \leq \lfloor \overline{x}_j \rfloor\},$$
$$\mathscr{F}_2 = \mathscr{F} \cap \{x : x_j \geq \lceil \overline{x}_j \rceil\}.$$

Of course, futher down the tree we can use the same branching rule to subdivide any $\mathscr{F}_j$ thus generated. In our case this leads to the two subproblems

$$(\mathscr{F}_1) \quad z = \max 4x_1 - x_2 \quad \text{s.t. } 7x_1 - 2x_2 \leq 14,\ x_2 \leq 3,\ 2x_1 - 2x_2 \leq 3,\ x_1 \leq 2,\ x \in \mathbb{Z}^2_+,$$

$$(\mathscr{F}_2) \quad z = \max 4x_1 - x_2 \quad \text{s.t. } 7x_1 - 2x_2 \leq 14,\ x_2 \leq 3,\ 2x_1 - 2x_2 \leq 3,\ x_1 \geq 3,\ x \in \mathbb{Z}^2_+.$$

So far we obtained the partial enumeration tree of Figure 21. Nodes $\mathscr{F}_1$ and $\mathscr{F}_2$ still need to be explored. We mark these nodes as *active*. The node $\mathscr{F}$ has been processed and is *inactive*.

Step 3: Pick an active node. We pick active node $(\mathscr{F}_1)$.

Note: $\mathscr{P}_1 \subsetneq \mathscr{P}$ and $\overline{x} \neq \mathscr{P}_1$ implies

$$\overline{z}_1 = \max\{c^{\mathrm{T}}x : x \in \mathscr{P}_1\} \leq \overline{z},$$

and the inequality is strict in general.

More generally, upon solving the LP relaxations of the subproblems obtained by up- and down-branching of a fractional variable, we will have

$$\max\{\overline{z}^{[1]}, \overline{z}^{[2]}\} \leq \overline{z},$$

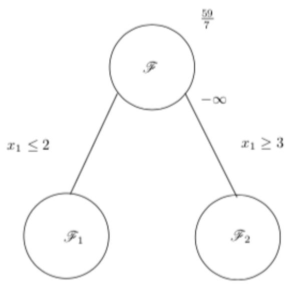


Figure 21: The partial enumeration tree.

since said variable would have to be allowed to take the fractional value for $\overline{z}$ to be attained as objective value. This will ensure that $\overline{z}$ decreases.

Step 4: Bounding $(\mathscr{F}_1)$**.** Need to solve the LP relaxation

$$(\mathscr{P}_1)\quad \overline{z}_1 = \max 4x_1 - x_2$$
$$\text{s.t. } 7x_1 - 2x_2 \leq 14,\ x_2 \leq 3,\ 2x_1 - 2x_2 \leq 3$$
$$x_1 \leq 2,\ x \geq 0.$$

Can we use a warm start?

$$(\mathscr{P})\quad \overline{x} = \arg\max c^{\mathrm{T}}x \quad \text{s.t. } Ax \leq b,\ x \geq 0 \qquad\qquad (\mathscr{D})\quad \overline{y} = \arg\min b^{\mathrm{T}}y \quad \text{s.t. } A^{\mathrm{T}}y \geq c,\ y \geq 0$$

$$\downarrow \qquad\qquad\qquad\qquad \downarrow$$

$$(\mathscr{P}_1)\quad \max c^{\mathrm{T}}x \quad \text{s.t. } Ax \leq b,\ x_1 \leq 2,\ x \geq 0 \qquad\qquad (\mathscr{D}_1)\quad \min b^{\mathrm{T}}y \quad \text{s.t. } A^{\mathrm{T}}y \geq c,\ y_4 \geq 0,\ y \geq 0.$$

Note: $\overline{x}$ infeasible $\qquad\qquad (\overline{y}, y_4 = 0)$ feasible

Add new constraint to tableau (blue row, new slack variable corresponding to red column):

1	0	1/7	2/7	0	0	20/7
0	1	0	1	0	0	3
0	0	-2/7	10/7	1	0	23/7
1	0	0	0	0	1	2
0	0	-4/7	-1/7	0	0	-59/7

Eliminate basic variable x_1 in new row:

$$\begin{array}{cccccc|c}
1 & 0 & 1/7 & 2/7 & 0 & 0 & 20/7 \\
0 & 1 & 0 & 1 & 0 & 0 & 3 \\
0 & 0 & -2/7 & 10/7 & 1 & 0 & 23/7 \\
0 & 0 & -1/7 & -2/7 & 0 & 1 & -6/7 \\
\hline
0 & 0 & -4/7 & -1/7 & 0 & 0 & -59/7
\end{array}$$

Apply dual simplex step to blue row: $(-1/7)/(-2/7) < (-4/7)/(-1/7) \Rightarrow$ pivot on x_4

$$\begin{array}{cccccc|c}
1 & 0 & 1/7 & 2/7 & 0 & 0 & 20/7 \\
0 & 1 & 0 & 1 & 0 & 0 & 3 \\
0 & 0 & -2/7 & 10/7 & 1 & 0 & 23/7 \\
0 & 0 & -1/7 & -2/7 & 0 & 1 & -6/7 \\
\hline
0 & 0 & -4/7 & -1/7 & 0 & 0 & -59/7
\end{array}$$

$(-1/7)/(-2/7) < (-4/7)/(-1/7) \Rightarrow$ pivot on x_4, i.e. eliminate all but red coefficient in column 4:

$$\begin{array}{cccccc|c}
1 & 0 & 0 & 0 & 0 & 1 & 2 \\
0 & 1 & -1/2 & 0 & 0 & 7/2 & 0 \\
0 & 0 & -1 & 0 & 1 & 5 & -1 \\
0 & 0 & 1/2 & 1 & 0 & -7/2 & 3 \\
\hline
0 & 0 & -1/2 & 0 & 0 & -1/2 & -8
\end{array}$$

Take another dual simplex step, pivoting on Row 3:

$$\begin{array}{cccccc|c}
1 & 0 & 0 & 0 & 0 & 1 & 2 \\
0 & 1 & -1/2 & 0 & 0 & 7/2 & 0 \\
0 & 0 & -1 & 0 & 1 & 5 & -1 \\
0 & 0 & 1/2 & 1 & 0 & -7/2 & 3 \\
\hline
0 & 0 & -1/2 & 0 & 0 & -1/2 & -8
\end{array}$$

$$\begin{array}{cccccc|c}
1 & 0 & 0 & 0 & 0 & 1 & 2 \\
0 & 1 & 0 & 0 & -1/2 & 1 & 1/2 \\
0 & 0 & 1 & 0 & -1 & -5 & 1 \\
0 & 0 & 0 & 1 & 1/2 & -1 & 5/2 \\
\hline
0 & 0 & 0 & 0 & -1/2 & -6/2 & -15/2
\end{array}$$

The re-optimised tableau yields the optimal solution of $(\mathscr{P}_1)$,

$$\overline{z}^{[1]} = \frac{15}{2}, \qquad \big(\overline{x}_1^{[1]}, \overline{x}_2^{[1]}\big) = \big(2, \frac{1}{2}\big).$$

Step 5: Branching. Fractional branching yields

$$\mathscr{F}_{11} = \mathscr{F}_1 \cap \{x : x_2 \leq 0\} = \mathscr{F}_1 \cap \{x : x_2 = 0\} \quad (\text{since } x_2 \geq 0),$$
$$\mathscr{F}_{12} = \mathscr{F}_1 \cap \{x : x_2 \geq 1\}.$$

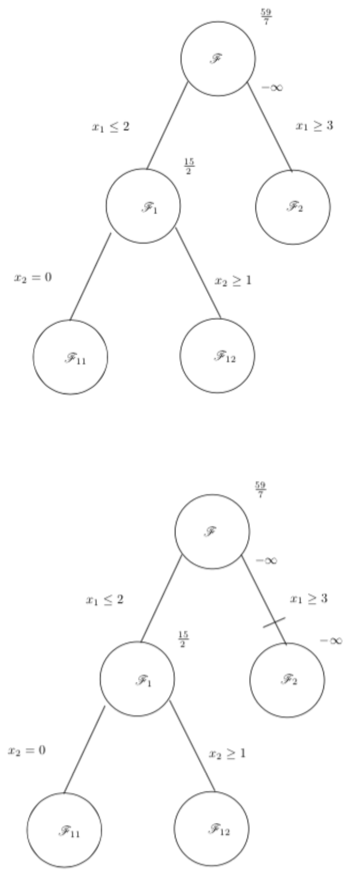


Figure 22: The updated partial enumeration tree.

We now arrive at the updated partial enumeration tree of Figure 8.2, and the new list of active nodes is $\mathscr{F}_{11}, \mathscr{F}_{12}, \mathscr{F}_2$.

Step 6. Choose the active node $\mathscr{F}_2$. Using dual simplex, solve LP relaxation $(\mathscr{P}_2)$. We find $\mathscr{P}_2 = \emptyset \Rightarrow \mathscr{F}_2 = \emptyset$. Prune by infeasibility.

Key Idea 8.3 (Pruning by bound)**.**

- *Nodes whose bounds certify that they do not contain an optimal solution need not be further processed.*
- *In particular, nodes with dual bound $-\infty$ need not be processed further (pruning by infeasibility).*

Step 7. We arbitrarily choose the active node $\mathscr{F}_{12}$ and sove the LP relaxation $(\mathscr{P}_{12})$ via dual simplex steps, yielding the optimal solution $(\overline{x}_1^{[12]}, \overline{x}_2^{[12]}) = (2, 1)$ and optimal value $\overline{z}^{[12]} = 7$.

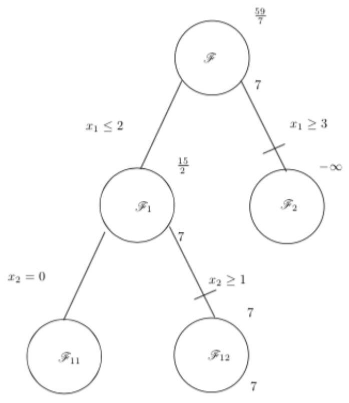


Figure 23: The partial enumeration tree.

Step 8. Since $\overline{x}^{[12]}$ is integral, this is a feasible solution for $(\mathscr{F}_1)$ and provides a lower bound $\underline{z}^{[12]} = 7$. In fact, $\mathscr{F}_{12}$ can now be pruned *by optimality*. The partial enumeration tree is now as in Figure 23.

Key Idea 8.4 (Pruning by optimality)**.** *Nodes whose bounds certify that their subproblem has been solved to optimality need not be processed any further.*

Step 9: Updating the incumbent. We store $(1, 2)$ as the best integer solution found so far and update the lower bounds $\underline{z} \leftarrow \max(\underline{z}, 7) = 7$, $\underline{z}^{[1]} \leftarrow \max(\underline{z}^{[1]}, 7) = 7$.

Step 10. Solve LP relaxation $(\mathscr{P}_{11})$ of only remaining active node $\mathscr{F}_{11}$. Dual simplex yields optimal solution $\left(\overline{x}_1^{[11]}, \overline{x}_2^{[11]}\right) = \left(\frac{3}{2}, 0\right)$ and optimal value $\overline{z}^{[11]} = 6$.

Since $\overline{z}^{[11]} < \underline{z}$, we can prune $\mathscr{F}_{11}$ *by bound* and arrive at the partial enumeration tree of Figure 24.

Step 11: Termination. There are no further active nodes left, and the algorithm terminates, returning the optimal solution $z = 7$ and the maximiser $x = (2, 1)$ that achieves it.

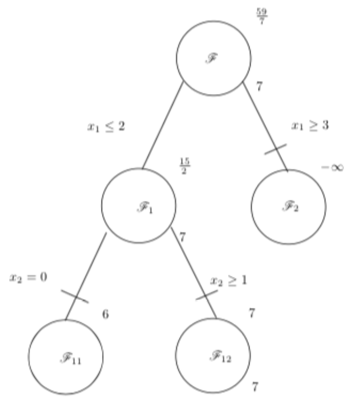


Figure 24: The final partial enumeration tree.

9 Lecture 9: More on Branch and Bound

9.1 General Branch-and-Bound Principles

Proposition 9.1 (Divide and conquer)**.** *Consider the problem*

$$z = \max\{c^{\mathrm{T}} x : x \in \mathscr{F}\},$$

where $\mathscr{F}$ denotes the set of feasible solutions (as we saw, $\mathscr{F}$ is usually defined implicitly via constraints). If $\mathscr{F}$ can be decomposed into a union of simpler sets $\mathscr{F} = \mathscr{F}_1 \cup \cdots \cup \mathscr{F}_k$ and if

$$z^{[j]} := \max\{c^{\mathrm{T}} x : x \in \mathscr{F}_j\} \quad (j = 1, \dots, k),$$

then $z = \max_j z^{[j]}$.

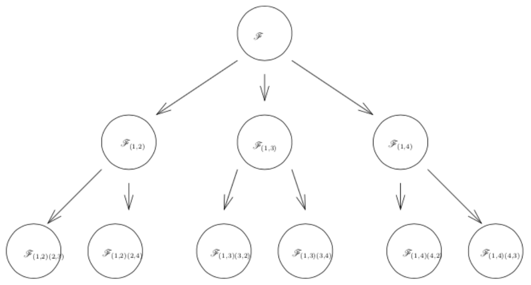
Proof. $\mathscr{F}$ is a relaxation of $\mathscr{F}_j$, so that $z^{[j]} \leq z$ for all j, and hence, $\max_j z^{[j]} \leq z$. Let x^* be optimal for the master problem, i.e., $x^* \in \mathscr{F}$ such that $z = c^{\mathrm{T}} x^*$. Then $x^* \in \mathscr{F}_i$ for some i, so that $z = c^{\mathrm{T}} x^* \leq z^{[i]} \leq \max_j z^{[j]}$. □

Example 9.2 (Enumeration tree)**.** *Let $\mathscr{F}$ be the set of feasible tours of the travelling salesman problem on a network of 4 cities. Let node 1 be the departure city.*

$\mathscr{F}$ can be subdivided $\mathscr{F} = \mathscr{F}_{(1,2)} \cup \mathscr{F}_{(1,3)} \cup \mathscr{F}_{(1,4)}$ into the disjoint sets of tours that start with an arc $(1,2)$, $(1,3)$ or $(1,4)$ respectively.

Each of the sets $\mathscr{F}_{(1,2)}$, $\mathscr{F}_{(1,3)}$ and $\mathscr{F}_{(1,4)}$ can be further subdivided according to the choice of the second arc, $\mathscr{F}_{(1,2)} = \mathscr{F}_{(1,2)(2,3)} \cup \mathscr{F}_{(1,2)(2,4)}$ etc.

Finally, we see that each of these sets corresponds to a specific TSP tour and cannot be further subdivided. We have found the enumeration tree *of the TSP tours depicted in Figure 25.*



$\mathscr{F}$
$\mathscr{F}_{(1,2)}$
$\mathscr{F}_{(1,3)}$
$\mathscr{F}_{(1,4)}$
$\mathscr{F}_{(1,2)(2,3)}$
$\mathscr{F}_{(1,2)(2,4)}$
$\mathscr{F}_{(1,3)(3,2)}$
$\mathscr{F}_{(1,3)(3,4)}$
$\mathscr{F}_{(1,4)(4,2)}$
$\mathscr{F}_{(1,4)(4,3)}$

Figure 25: The enumeration tree of the TSP example.

Proposition 9.3 (Bound propagation)**.** *Consider the problem*

$$z = \max\{c^{\mathrm{T}}x : x \in \mathscr{F}\},$$

and let $\mathscr{F} = \mathscr{F}_1 \cup \cdots \cup \mathscr{F}_k$ be a decomposition of its feasible domain into smaller sets. Let $\underline{z}^{[j]} \leq z^{[j]} \leq \overline{z}^{[j]}$ be lower and upper bounds on $z^{[j]} = \max\{c^{\mathrm{T}}x : x \in \mathscr{F}_j\}$ for all j. Then

$$\underline{z} := \max_j \underline{z}^{[j]} \leq z \leq \max_j \overline{z}^{[j]} =: \overline{z}$$

gives an upper and lower bound on z.

Proof. Since $(\mathscr{F})$ is a relaxation of $(\mathscr{F}_j)$, we have $\underline{z}^{[j]} \leq z^{[j]} \leq z$ for all j, and hence,

$$\max_j \underline{z}^{[j]} \leq z.$$

On the other hand, by Proposition (Divide and conquer), we have

$$z = \max_j z^{[j]} \leq \max_j \overline{z}^{[j]}.$$

□

Proposition 9.4 (Pruning by bound)**.** *A branch $\mathscr{F}_j$ can be pruned when $\overline{z}^{[j]} \leq \underline{z}$.*

Proof. By construction of $\underline{z}$, the incumbent x^* (our best solution for the root problem $(\mathscr{F})$ thus far encountered) satisfies

$$c^{\mathrm{T}}x^* = \underline{z} \geq \overline{z}^{[j]} \geq c^{\mathrm{T}}x \quad \forall\, x \in \mathscr{F}_j.$$

Therefore, the incumbent cannot be improved by searching over $\mathscr{F}_j$. □

Proposition 9.5 (Pruning by infeasibility)**.** *If $\mathscr{F}_j = \emptyset$, then the corresponding branch can be pruned.*

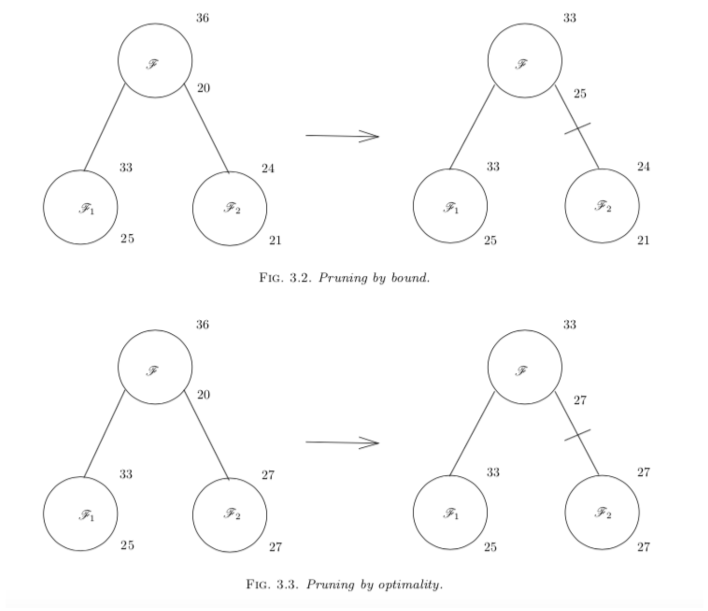


Figure 26: Pruning in action.

Proof. This is a special case of pruning by bound in which $\overline{z}^{[j]} = -\infty$. □

Remark 9.6. *It may not be obvious that $\mathscr{F}_j$ is empty, but this may be algorithmically detected, e.g., by LP relaxation.*

Proposition 9.7 (Pruning by optimality)**.** *When $\underline{z}^{[j]} = \overline{z}^{[j]}$ for some j, then the branch corresponding to $\mathscr{F}_j$ needs no further consideration.*

Proof. $z^{[j]} = \underline{z}^{[j]} = \overline{z}^{[j]}$ certfies that we have solved this branch to optimality. The optimal point of the branch has automatically become the incumbent if it is the best solution found so far for the root problem. □

9.2 The General Branch & Bound Framework

Data Specification 9.8 (General B&B)**.**

```
// input
objective $x \mapsto f(x)$;
implicit description of feasible set $x \in \mathscr{F}$ of root problem
$z = \max\{f(x) : x \in \mathscr{F}\}$;
// working data
list AN of active nodes;
global primal bound $\underline{z} \le z$;
global dual bound $\overline{z} \ge z$;
incumbent $x^*$, i.e. best solution found so far;
subproblem primal bound $\underline{z}^{[j]} \le z^{[j]} := \max\{f(x) : x \in \mathscr{F}_j\}$;
subproblem dual bound $\overline{z}^{[j]} \ge z^{[j]}$;
// outputs
global bounds $\underline{z}, \overline{z}$ to provide certificate of optimality (when $\underline{z} = \overline{z}$) or
approximation guarantee;
incumbent $x^*$ as optimal solution (or solution with approximation guarantee
when $\underline{z} < \overline{z}$);
```

Methods 9.9 (General B&B)**.**

```
// required methods
method to compute dual bounds by solving an easy problem

    $(\mathscr{R}_j) \quad \overline{z}^{[j]} = \max\{g_j(x) : x \in \mathscr{R}_j\}$

e.g., $g_j(x) \ge f(x) \ \forall x \in \mathscr{F}_j$ and $\mathscr{R}_j \supset \mathscr{F}_j$ (relaxation);
branching rule to split $\mathscr{F}_j = \mathscr{F}_{j_1} \cup \dots \cup \mathscr{F}_{j_k}$ so that subproblems

    $(\mathscr{F}_j) \quad \max\{f(x) : x \in \mathscr{F}_j\}$

all fall into the same problem class as root problem $(\mathscr{F})$ to which above methods
apply;
// optional methods
heuristic to find $y \in \mathscr{F}_j$ and compute primal bounds $\underline{z}^{[j]} = f(y)$;
selection rule for $(\mathscr{F}_j) \in \mathbf{AN}$;
```

Algorithm 9.10 (General B&B).

```
$AN = \{(\mathscr{F})\}$; $\underline{z} = -\infty$; $\overline{z} = +\infty$; $x^* = NaN$; // initialisation
while $AN \neq \emptyset$ do
    choose $(\mathscr{F}_j) \in AN$;
    if $\overline{z}^{[j]} \leq \underline{z}$ then
        $AN := (AN \setminus \{(\mathscr{F}_j\}))$; // prune by bound
    else
        solve $x^{[j]} = \arg\max\{g_j(x) : x \in \mathscr{R}_j\}$; // relaxation of $(\mathscr{F}_j)$
        $\overline{z}^{[j]} := g_j(x^{[j]})$; // $\overline{z}^{[j]} := -\infty$ if $\mathscr{R}_j = \emptyset$
        $\overline{z} := \max\{\overline{z}^{[j]} : (\mathscr{F}_j) \in AN\}$; // update global upper
            bound
        if $x^{[j]} \in \mathscr{F}_j$ then
            $y^{[j]} := x^{[j]}$;
        else
            attempt to find $y^{[j]} \in \mathscr{F}_j$ via heuristic; // unassigned if
                unsuccessful
        end
        $\underline{z}^{[j]} := \max(\underline{z}^{[j]}, f(y^{[j]}))$; // set $f(y^{[j]}) := -\infty$ if $y^{[j]}$
            unassigned
        if $\underline{z}^{[j]} > \underline{z}$ then
            $x^* := y^{[j]}$; // update incumbent
            $\underline{z} := \underline{z}^{[j]}$; // update global lower bound
        end
        if $\underline{z}^{[j]} = \overline{z}^{[j]}$; then
            $AN := (AN \setminus \{\mathscr{F}_j\})$; // prune by optimality resp.
                infeasiblity
        else
            $AN \leftarrow (AN \setminus \{(\mathscr{F}_j)\}) \cup \{(\mathscr{F}_{j_1}), \dots, (\mathscr{F}_{j_k})\}$; // branching
            $\overline{z}^{[j_\ell]} := \overline{z}^{[j]}$, $(\ell = 1, \dots, k)$; // child nodes inherit
                dual bound
        end
    end
end
```

9.3 LP Based Branch and Bound

Data Specification 9.11 (LP based B&B).

```
// input
objective $x \mapsto c^{\mathrm{T}}x$;
description of $\mathscr{F}$ via formulation $\mathscr{P}$ (polyhedron);
```

Methods 9.12 (LP based B&B).

```
// required methods
compute dual bounds $\overline{z}^{[j]} = c^{\mathrm{T}} x^{[j]}$ by solving LP relaxation
    $(\mathscr{P}_j) \quad x^{[j]} = \arg\max\{c^{\mathrm{T}} x : x \in \mathscr{P}_j\};$
branch on fractional variable $x_i^{[j]} \notin \mathbb{Z}$
    $\mathscr{P}_{j_1} = \mathscr{P}_j \cap \left\{x : x_i \le \lfloor x_i^{[j]} \rfloor \right\};$
    $\mathscr{P}_{j_2} = \mathscr{P}_j \cap \left\{x : x_i \ge \lceil x_i^{[j]} \rceil \right\};$
```

9.4 Adaptation to Specific Problems

Branch & Bound is a framework that can be used to build customised algorithms for specific problem classes:

- choice of algorithm to compute dual bounds,
- choice of heuristics to compute primal bounds,
- branching rules,
- node selection rules,
- preprocessing,
- combination with cutting planes (see later lectures).

9.4.1 Choice of Algorithm to Compute Dual Bounds

For most IPs, LP-based B&B is significantly faster when the simplex algorithm is applied to the dual of the LP relaxations $(\mathscr{P}_j)$ rather than on the primal problem.

To understand why, let us see what happens when an LP

$$\begin{aligned}(\mathscr{P}) \quad \max_{x\in\mathbb{R}^n} \; & c^{\mathrm{T}} x \\ \text{s.t. } & \mathbf{a}_i^{\mathrm{T}} x \le b_i, \quad (i = 1, \dots, m) \\ & x_j \ge 0, \quad (j = 1, \dots, n),\end{aligned}$$

where $\mathbf{a}_i^{\mathrm{T}}$ are row vectors, is amended by introducing a new constraint:

$$\begin{aligned}(\mathscr{P}') \quad \max_{x\in\mathbb{R}^n}\ & c^{\mathrm{T}}x\\ \text{s.t. } & \mathbf{a}_i^{\mathrm{T}}x \le b_i, \quad (i=1,\dots,m)\\ & \mathbf{a}_{m+1}^{\mathrm{T}}x \le b_{m+1},\\ & x_j \ge 0, \quad (j=1,\dots,n).\end{aligned}$$

An optimal basic feasible solution (x_B, x_N) of $(\mathscr{P})$ is not necessarily feasible for $(\mathscr{P})'$, which hinders us from re-optimising the solution via primal simplex pivots.

The change in the dual

$$\begin{aligned}(\mathscr{D}) \quad \min_{y\in\mathbb{R}^m}\ & b^{\mathrm{T}}y\\ \text{s.t. } & \sum_{i=1}^m y_i\mathbf{a}_i \ge c,\\ & y_i \ge 0, \quad (i=1,\dots,m)\end{aligned}$$

is more benign, as the amendment of $(\mathscr{P})$ into $(\mathscr{P})'$ corresponds to the introduction of a new variable y_{m+1},

$$\begin{aligned}(\mathscr{D}') \quad \min_{y\in\mathbb{R}^{m+1}}\ & b^{\mathrm{T}}y\\ \text{s.t. } & \sum_{i=1}^m y_i\mathbf{a}_i + y_{m+1}\mathbf{a}_{m+1} \ge c,\\ & y_i \ge 0, \quad (i=1,\dots,m+1)\end{aligned}$$

If (y_B, y_N) is an optimal basic feasible solution of $(\mathscr{D})$, then setting $y_{m+1} = 0$ and adding it to the set of non-basic variables yields a basic feasible solution for $(\mathscr{D}')$ that can be re-optimised via a few extra simplex pivots. $\Rightarrow$ Use Dual Simplex Method.

Special Cases: The simplex algorithm is not always the algorithm of choice in LP-based B&B.

Example 9.13 (Greedy solution of knapsack LP). *Consider the 0-1 knapsack problem*

$$\max\left\{\sum_{j=1}^n c_jx_j : \sum_{j=1}^n a_jx_j \le b,\ x\in\mathbb{B}^n\right\},$$

where $a_j, c_j > 0$ for $j=1,\dots,n$. The LP relaxation

$$\max\left\{\sum_{j=1}^n c_jx_j : \sum_{j=1}^n a_jx_j \le b,\ 0\le x_j\le 1, \quad (j=1,\dots,n)\right\}$$

has special structure that makes it possible to solve it greedily:

Re-index the variables to that

$$\frac{c_1}{a_1} \geq \cdots \geq \frac{c_n}{a_n} > 0,$$
$$\sum_{j=1}^{r-1} a_j \leq b, \quad \sum_{j=1}^{r} a_j > b.$$

The optimal solution of the LP relaxation is then given by $x_j = 1$ for $j = 1, \dots, r-1$, $x_r = (b - \sum_{j=1}^{r-1} a_j)/a_r$ and $x_j = 0$ for $j > r$. (See problem sheet.)

9.4.2 Heuristics to Compute Primal Bounds

Example 9.14 (Knapsack primal bounds)**.** *Consider the 0-1 knapsack problem*

$$\begin{aligned} z^* = \max_x \sum_{j=1}^{n} c_j x_j \\ \textit{subject to} \sum_{j=1}^{n} a_j x_j \leq b, \\ x_j \in \{0,1\}, \quad (j = 1, \dots, n), \end{aligned}$$

where a_j $(j = 1, \dots, n)$ and b are positive integers, and c_j are rational numbers.

Let z_{LP} be the optimal objective value of the LP-relaxation and z_{gh} the primal bound obtained from the greedy heuristic described on the next slide. Let $r - 1 = \max\{i : \sum_{j=1}^{i} a_j \leq b\}$ and $\xi = b - \sum_{j=1}^{r-1} a_j$.

Then the following approximation bound holds,

$$z_{LP} \geq z^* \geq z_{gh} \geq \left(1 - \frac{\xi}{b}\right) \times z_{LP}.$$

Algorithm 9.15 (Greedy knapsack heuristic)**.**

```
Re-index s.t. $c_1/a_1 \geq c_2/a_2 \geq \cdots \geq c_n/a_n$. Set $z = 0$, $v = b$;
    // initialisation
for j=1,...,n do
    if $a_j \leq v$ then
        $x_j = 1$;
        $v \leftarrow v - a_j$;
        $z = z + c_j$;
    else
        $x_j = 0$;
    end
end
```

9.4.3 Branching Rules

Branching on most fractional variable. Let C be the set of fractional variables of the solution x^* to a LP relaxation. The *most fractional variable* approach is to branch on the variable that corresponds to the index

$$j = \arg\max_{i\in C} \min\{f_i, 1-f_i\},$$

where $f_i = x_i^* - \lfloor x_i^* \rfloor$.

Branching by priorities. In this approach the user can indicate a priority of importance for the decision variables to be integer. The system will then branch on the fractional variable with highest priority.

Example 9.16. *Fixed charge network Consider a fixed charge network problem*

$$\min\{c^{\mathrm{T}}x + fy : Nx = b,\, x \le uy,\, x \in \mathbb{R}^n_+,\, y \in \mathbb{Z}^n_+\},$$

where N is the node-arc incidence matrix of the network, and b is the demand vector.

Since rounding up the variables y_j corresponding to large fixed costs f_j changes the objective function more severely than rounding y_j corresponding to small fixed costs, we prioritise the y_j in order of decreasing fixed costs f_j.

GUB/SOS branching. Many IP models contain *generalised upper bound* (GUB) or *special ordered sets* (SOS) constraints of the form

$$\sum_{j=1}^{k} x_j = 1,$$

with $x_j \in \mathbb{B}$ for all j. In this case it is not good to branch on a factional variable x_i^* from the solution x^* of a LP relaxation, since the branching does not lead to balanced sets:

$$S_1 = \{x \in S : x_i = 1\}$$

contains only one point $x_i = 1$, $x_j = 0\, \forall\, j \neq i$, whereas

$$S_2 = \{x \in S : x_i = 0\}$$

contains $|S| - 1$ points.

In these cases, a better choice of branching is given by fixing an order $j_1, \dots, j_k$ of the variables and choosing

$$S_1 = S \cap \{x : x_{j_i} = 0,\, i = 1, \dots, r\},$$
$$S_2 = S \cap \{x : x_{j_i} = 0,\, i = r+1, \dots, k\},$$

where $r = \min\big\{s : \sum_{i=1}^{s} x_{j_i}^* \ge \frac{1}{2}\big\}$.

Strong Branching. This is used only on difficult problems where it is worthwhile spending more time to find a good branching. In this approach one

i) chooses a set C of candidate variables, branches up and down for each $x_j \in C$,

ii) computes upper bounds z_j^U and z_j^D by solving the LP relaxations of the up and down branching corresponding to x_j,

iii) chooses the variable having the largest effect

$$j^* = \arg\min\{\max(z_j^U, z_j^D) : j \in C\}$$

as the *actual branching variable* for the branch-and-bound scheme.

That is, one explores several possible branchings and chooses to pursue the branches below only the most promising branching variable.

9.4.4 Node Selection

Depth-First. A *depth-first strategy* aims at finding a feasible solution quickly, by only choosing an active node that is a direct descendant of the previously processed node. It is implemented by operating a *last-in-first-out* stack for the active nodes.

Best-Node-First. To minimise the total number of nodes processed during the run of the algorithm, the optimal strategy is to always choose the node with the largest upper bound, i.e., S_j such that

$$\overline{z}^{[j]} = \max\{\overline{z}^{[i]} : S_i \in \mathrm{AN}\}.$$

Under this strategy we will never branch on a node S_t whose upper bound $\overline{z}^{[t]}$ is smaller than the optimal value z of S. This is called a *best-node-first* strategy.

The depth-first and best-node-first strategies are usually mutually contradictory, so a compromise has to be reached. Usually, depth-first is used initially until a feasible solution is found and a lower bound $\underline{z}$ is established.

10 Lecture 10: Delayed Column Generation

10.1 The Dantzig-Wolfe Reformulation

We will now explore a technique that can be applied to extremely large integer programming problems of *block angular* form

$$\begin{aligned}
\text{(IP)} \quad z = \max & \sum_{k=1}^{K} c^{k\,\mathrm{T}} x^k \\
\text{s.t.} & \sum_{k=1}^{K} A^k x^k = b \\
& x^k \in X^k = \{x^k \in \mathbb{Z}_+^{n_k} : D^k x^k \le d^k\}, \quad (k = 1, \dots, K).
\end{aligned}$$

The key features of such problems are the following:

- The decision vector $x = (x^1, \dots, x^K)$ is partitioned into K blocks $x^k \in \mathbb{R}^{n_k}$, $(k = 1, \dots, K$.
- The individual blocks are linked to one another only via the *joint* constraints $\sum_{k=1}^{K} A^k x^k = b$, while the remaining constraints decompose.
- We assume that each $X^k = \{x^{k,t}\}_{t=1}^{T_k}$ consists of a large but finite number of points, so that we can write

$$X^k = \left\{ x^k \in \mathbb{R}^{n_k} : x^k = \sum_{t=1}^{T_k} \lambda_{k,t} x^{k,t}, \sum_{t=1}^{T_k} \lambda_{k,t} = 1, \lambda_{k,t} \in \{0,1\}, (t = 1, \dots, T_k) \right\}.$$

Example 10.1 (Uncapacitated facility location)**.** *The Uncapacitated Facility Location Problem from Lecture 1 is of this form,*

$$\begin{aligned}
\textit{(UFL)} \quad z = \min & \sum_{k=1}^{n} \left[\sum_{i=1}^{m} c_{ik} x_{ik} + f_k y_k \right] \\
\textit{s.t.} & \sum_{k=1}^{n} x_{ik} = 1 \quad (i = 1, \dots, m) \\
& x_{ik} - y_k \le 0 \quad (i = 1, \dots, m; k = 1, \dots, n) \qquad \textit{("strong formulation")} \\
& x \in \mathbb{R}_+^{m \times n}, \ y \in \{0,1\}^n,
\end{aligned}$$

- $x^k = (x_{1k}, \dots, x_{mk}, y_k)$ *are the blocks of variables that deal with the proportion* x_{ik} *of customer i's demand to meet from a fixed location k,*
- *the flag variable* y_k *that indicates whether facility k is in operation,*

- *the constraints $\sum_{k=1}^{n} x_{ik} = 1$ $(i = 1, \dots, m)$ that link each fixed customer to decisions at all facilities play the role of the joint constraints*

$$\begin{bmatrix} \mathrm{I}_m & 0_{m\times 1} & \mathrm{I}_m & 0_{m\times 1} & \dots & \mathrm{I}_m & 0_{m\times 1} \end{bmatrix} \begin{bmatrix} x^1 \\ \vdots \\ x^n \end{bmatrix} = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix},$$

that is, $K = n$ and

$$A_k = \begin{bmatrix} \mathrm{I}_m & 0_{m\times 1} \end{bmatrix}, \quad (k = 1, \dots, K).$$

The *Dantzig-Wolfe Reformulation* exploits the representations

$$X^k = \left\{ x^k \in \mathbb{R}^{n_k} : x^k = \sum_{t=1}^{T_k} \lambda_{k,t} x^{k,t}, \sum_{t=1}^{T_k} \lambda_{k,t} = 1, \lambda_{k,t} \in \{0,1\} \, (t = 1, \dots, T_k) \right\},$$

which are available in theory but generally not in practice, to make it possible to use the $\lambda_{k,t}$ as decision variables.

Substituting $\sum_{t=1}^{T_k} \lambda_{k,t} x^{k,t}$ for x^k in (IP), we obtain the *IP Master Problem*

$$\begin{aligned} \text{(IPM)} \quad z = \max & \sum_{k=1}^{K} \sum_{t=1}^{T_k} (c^{k\,\mathrm{T}} x^{k,t}) \lambda_{k,t} \\ \text{s.t.} & \sum_{k=1}^{K} \sum_{t=1}^{T_k} (A^k x^{k,t}) \lambda_{k,t} = b, \\ & \sum_{t=1}^{T_k} \lambda_{k,t} = 1, \quad (k = 1, \dots, K), \\ & \lambda_{k,t} \in \{0,1\}, \quad (t = 1, \dots, T_k; k = 1, \dots, K). \end{aligned}$$

This IP is of course equivalent to the original problem (IP), but inconveniently, it has many more variables, though on the plus side it has far fewer constraints.

Example 10.2 (UFL continued)**.** *Let us consider UFL again, this time in the form of the "weak" formulation*

$$\begin{aligned} \textit{(UFLW)} \quad z = \min & \sum_{k=1}^{n} \begin{bmatrix} c_{1k} & \dots & c_{mk} & f_k \end{bmatrix} x^k \\ \textit{s.t.} & \sum_{k=1}^{n} x_{ik} = 1 \quad (i = 1, \dots, m) \\ \textit{for } (k = 1, \dots, n), & \begin{cases} \sum_{i=1}^{m} x_{ik} \le m y_k & \textit{("weak formulation")} \\ x^k = \begin{bmatrix} x_{1k} & \dots & x_{mk} & y_k \end{bmatrix}^{\mathrm{T}} \in \{0,1\}^{m+1}. \end{cases} \end{aligned}$$

We have

$$\begin{aligned} X^k &= \left\{ x^k : \sum_{i=1}^m x_{ik} \leq m y_k,\ x_{ik} \in \{0,1\},\ (i = 1, \dots, m),\ y_k \in \{0,1\} \right\} \\ &= \left\{ (x^k_S, 1) \right\}_{S \subseteq M} \cup \left\{ (\mathbf{0}, 0) \right\}, \end{aligned}$$

where x^k_S is the incidence vector of $S \subseteq M = \{1, \dots, m\}$ (which we associate with the variable λ^k_S), and $(\mathbf{0}, 0)$ is the zero vector (which we associate with the variable ν^k).

Note that in this case all X^k are the same set $(k = 1, \dots, n)$ with $T_k = 2^m$ elements, and if $x^{k,t} = (x^k_S, 1)$, then $A^k x^{k,t} = \mathrm{I}_m\, x^k_S = x^k_S$.

The IP Master Problem is the following,

$$\begin{aligned} (IPM) \quad \min\ & \sum_{k=1}^n \left(\sum_{S \neq \emptyset} \left(\sum_{i \in S} c_{ik} + f_k \right) \lambda^k_S + f_k \lambda^k_\emptyset \right) \\ \text{s.t. } & \sum_{k=1}^n \sum_{S \subseteq M : i \in S} \lambda^k_S = 1, \quad (i = 1, \dots, m) \\ & \sum_{S \neq \emptyset} \lambda^k_S + \lambda^k_\emptyset + \nu^k = 1, \quad (k = 1, \dots, n) \\ & \lambda^k_S,\ \nu^k \in \{0,1\}, \quad (S \subseteq M; k = 1, \dots, n). \end{aligned}$$

Since $f_k > 0$, a solution could not be optimal unless $\lambda^k_\emptyset = 0\ \forall k$, so that (IPM) is equivalent to

$$\begin{aligned} (IPM) \quad \min\ & \sum_{k=1}^n \sum_{S \neq \emptyset} \left(\sum_{i \in S} c_{ik} + f_k \right) \lambda^k_S \\ \text{s.t. } & \sum_{k=1}^n \sum_{S \subseteq M : i \in S} \lambda^k_S = 1, \quad (i = 1, \dots, m) \\ & \sum_{S \neq \emptyset} \lambda^k_S \leq 1, \quad (j = 1, \dots, n) \\ & \lambda^k_S \in \{0,1\}, \quad (\emptyset \neq S \subseteq M; j = 1, \dots, n). \end{aligned}$$

Example 10.3 (Symmetric travelling salesman problem)**.** *A special case in which $K = 1$ occurs for the STSP on a graph $G = (V, E)$, which can be formulated as follows,*

$$\begin{aligned} \min\ & \sum_{e \in E} c_e x_e \\ \text{s.t. } & \sum_{e \in \delta(i)} x_e = 2, \quad (i \in N), \\ & x \in X^1, \end{aligned}$$

where N is the set of nodes (cities), $\delta(i)$ the set of edges incident to i, x_e the indicator variable for edge e, and X^1 the set of indicator vectors of 1-trees of the graph.

A 1-tree is a subset $E' \subset E$ of edges of which exactly two are incident to node 1, that is,

$$|E' \cap \delta(1)| = 2,$$

and $E' \setminus \delta(1)$ is a spanning tree on the remaining nodes $2, \dots, n$, and hence it is of cardinality $n-2$.

Every Hamiltonian tour is a 1-tree that satisfies the degree constraint $|E' \cap \delta(i)| = 2$ for all nodes $i \in N$. Conversely, every 1-tree that satisfies these degree constraints is a Hamiltonian tour.

We use a conceptual enumeration $\{E^t : t = 1, \dots, T^1\}$ of the set of 1-trees and write

$$x_e = \sum_{t:\, e \in E^t} \lambda_t,$$

subject to $\lambda_t \in \{0,1\}$ for all t and $\sum_{t=1}^{T_1} \lambda_t = 1$. This defines the indicator vector of the single 1-tree for which $\lambda^t = 1$.

The degree constraints become

$$\sum_{e \in \delta(i)} x_e = \sum_{e \in \delta(i)} \sum_{t:\, e \in E^t} \lambda_t = \sum_t d_i^t \lambda_t = 2,$$

where d_i^t is the degree of node i in the 1-tree E^t. This yields the IP Master Problem

$$\begin{aligned} \min & \sum_{t=1}^{T_1} \left(\sum_{e \in E} c_e x_e^t \right) \lambda_t \\ \text{s.t.} & \sum_{t=1}^{T_1} d_i^t \lambda_t = 2, \quad (i \in N) \\ & \sum_{t=1}^{T_1} \lambda_t = 1, \\ & \lambda_t \in \{0,1\}, \quad (t = 1, \dots, T_1). \end{aligned}$$

10.2 LP Master Problem

To recap so far, starting with an IP of the form

$$\begin{aligned} \text{(IP)} \quad z = \max_x & \sum_{k=1}^{K} c^{k\,\mathrm{T}} x^k \\ \text{s.t.} & \sum_{k=1}^{K} A^k x^k = b \\ & x^k \in X^k = \{x^k \in \mathbb{Z}_+^{n_k} : D^k x^k \le d^k\}, \quad (k = 1, \dots, K), \end{aligned}$$

we have derived the *IP Master Problem*

$$
\begin{aligned}
\text{(IPM)}\quad z = \max_{\lambda} & \sum_{k=1}^{K}\sum_{t=1}^{T_k}(c^{k\,\mathrm{T}}x^{k,t})\lambda_{k,t}\\
\text{s.t.} & \sum_{k=1}^{K}\sum_{t=1}^{T_k}(A^k x^{k,t})\lambda_{k,t} = b,\\
& \sum_{t=1}^{T_k}\lambda_{k,t} = 1, \quad (k = 1,\dots,K),\\
& \lambda_{k,t}\in\{0,1\}, \quad (t = 1,\dots,T_k; k = 1,\dots,K).
\end{aligned}
$$

Next, we consider the LP relaxation of (IPM), called the *LP Master Problem*,

$$
\begin{aligned}
\text{(LPM)}\quad z^{LPM} = \max_{\lambda} & \sum_{k=1}^{K}\sum_{t=1}^{T_k}(c^{k\,\mathrm{T}}x^{k,t})\lambda_{k,t}\\
\text{s.t.} & \sum_{k=1}^{K}\sum_{t=1}^{T_k}(A^k x^{k,t})\lambda_{k,t} = b, && (22)\\
& \sum_{t=1}^{T_k}\lambda_{k,t} = 1, \quad (k = 1,\dots,K), && (23)\\
& \lambda_{k,t}\ge 0, \quad (t = 1,\dots,T_k; k = 1,\dots,K), && (24)
\end{aligned}
$$

in which there is a column

$$
\begin{bmatrix} c^{k\,\mathrm{T}}x^{k,t} \\ A^k x^{k,t} \\ e_k \end{bmatrix}
$$

for every $x^{k,t}\in X^k$ $(t = 1,\dots,T_k)$ and every $(k = 1,\dots,K)$, where e_k is the k-th canonical unit vector in $\mathbb{R}^K$. Equations (23) are called *convexity constraints*.

Associating (23) with dual variables $\{\mu_k\}_{k=1}^K$, and (22) with dual variables $\{\pi_i\}_{i=1}^m$, the dual of (LPM) is the following problem,

$$
\begin{aligned}
(DM)\quad z^{DM} = \min_{\mu,\pi} & \sum_{i=1}^{m} b_i\pi_i + \sum_{k=1}^{K}\mu_k\\
\text{s.t. } & \pi^{\mathrm{T}}A^k x^k + \mu_k \ge c^{k\,\mathrm{T}}x^k, \quad (x^k\in X^k).
\end{aligned}
$$

We now wish to apply the simplex algorithm to (LPM), but this is impossible because there are too many columns in this LP.

To get around this problem, we consider *Restricted LP Master Problems*

$$
\begin{aligned}
\text{(RM)}\quad \tilde z^{RM} = {} & \max \tilde c^{\mathrm{T}}\tilde\lambda\\
& \text{s.t. } \tilde A\tilde\lambda = \tilde b,\\
& \tilde\lambda \ge 0
\end{aligned}
$$

that arise from (LPM) by retaining only a subset of columns, *at least one from each k*.

Conceptually, this is the same as fixing $\lambda_{k,t} = 0$ for (k, t) that do not correspond to one of the chosen columns.

10.3 Delayed Column Generation

This is a simplex variant for solving LP Master Problem based on the following ideas:

- Use the simplex algorithm to find an optimal solution of a given reduced master problem (RM).
- Use this optimal solution to identify a further column to add (if necessary) and generate a new reduced master problem (RM$_+$).
- Solve (RM$_+$) via warm-start simplex.

Thus, instead of generating all the problem data of (LPM) initially, data is generated as and when it is needed. Typically, (LPM) is solved long before all the columns are generated!

Let us now discuss the details of this approach. We start with a reduced master problem

$$\begin{aligned} \text{(RM)} \quad \tilde{z}^{RM} &= \max \tilde{c}^{\mathrm{T}} \tilde{\lambda} \\ \text{s.t. } \tilde{A}\tilde{\lambda} &= \tilde{b}, \\ \tilde{\lambda} &\geq 0 \end{aligned}$$

and solve for an optimal solution $\tilde{\lambda}^*$ and an optimal dual solution $(\pi, \mu) \in \mathbb{R}^m \times \mathbb{R}^K$.

Note that $\tilde{\lambda}^*$ can be extended to a primal feasible solution of (LPM) by setting the $\tilde{\lambda}_{k,t} = 0$ for (k, t) outside the subset of columns chosen to form (RM).

Therefore, we have

$$\tilde{z}^{RM} = \tilde{c}^{\mathrm{T}} \tilde{\lambda}^* = \sum_{i=1}^{m} \pi_i b_i + \sum_{k=1}^{K} \mu_k \leq z^{LPM}.$$

Next, we check whether (π, μ) is dual feasible (feasible for (DM)), that is, whether

$$c^{k\,\mathrm{T}} x - \pi^{\mathrm{T}} A^k x - \mu_k \leq 0, \quad (x \in X^k;\ k = 1, \ldots, K).$$

This can be checked by solving an optimisation problem over each X^k,

$$(\mathrm{CGIP}_k) \quad \tilde{x}^k = \arg\max_x \, (c^k - \pi^{\mathrm{T}} A^k)x - \mu_k \quad \text{s.t. } x \in X^k,$$

and by checking whether $\zeta_k := (c^k - \pi^{\mathrm{T}} A^k)\tilde{x}^k - \mu_k \leq 0$.

Note that (CGIP$_k$) is an IP problem with only n_k integer decision variables, whereas the original IP had $\sum_{k=1}^K n_k$ integer decision variables.

Keep in mind that (CGIP$_k$) needs to be solved for $k = 1, \dots, K$. Thus, in solving these subproblems, the original master problem decomposes into subproblems that can be solved in parallel!

Decision to terminate or add a new column:

- If $\zeta_k \leq 0$ for all k, then $\tilde{\lambda}^*$ is optimal for (LPM): since (π, μ) is dual feasible, we have

  $$z^{LPM} \geq \tilde{z}^{RM} = \tilde{c}\tilde{\lambda}^* = \sum_{i=1}^m \pi_i b_i + \sum_{k=1}^K \mu_k \geq z^{DM} = z^{LPM},$$

  and equality must hold throughout. Thus, the algorithm can be stopped in this case.

- Otherwise, there exists k such that $\zeta_k > 0$. In this case, adding the column

  $$\begin{bmatrix} c^k \tilde{x}^k \\ A^k \tilde{x}^k \\ e_k \end{bmatrix}$$

  to (RLPM) leads to a new restricted LP master problem (RLPM$_+$) that can be reoptimised using warmstarting.

Note that solving the subproblems (CGIP$_k$) automatically takes care of generating a new column on which we are guaranteed to make progress toward solving Problem (LPM).

Discussion:

By iteratively adding columns as described above, we ensure that the algorithm behaves like the primal simplex applied to (LPM) with a legitimate sequence of pivots, but only the data of columns that are actually needed in pivots is ever generated.

This leads to a massive reduction in memory requirements.

However, the reduced problem (RLPM) grows over time, unless columns that leave the basis are discarded. At some stage this becomes a memory problem and the algorithm may have to be stopped before finding an optimal solution to (LPM).

We are then interested in knowing how closely the final primal bound $\underline{z} := z^{RM}$ approximates z^{LPM}. To this end, a *dual bound* $\overline{z} \geq z^{LPM}$ is required.

The primal and dual bounds $\underline{z}$ and $\overline{z}$ are also extremely useful when applying column generation in the context of branch-and-bound.

Calculating Dual Bounds:

A dual bound is obtained by recycling the work we have already done:

By construction of ζ_k we have

$$(c^k - \pi A^k)x - \mu_k - \zeta_k \leq 0, \quad (x \in X^k),$$

which shows that $(\pi, \mu + \zeta)$ is dual feasible for (LPM). Therefore,

$$\overline{z} := \pi b + \sum_{k=1}^{K} \mu_k + \sum_{k=1}^{K} \zeta_k$$

is a valid upper bound.

11 Lecture 11: Branch and Price

11.1 Recap on Delayed Column Generation

We can summarise the delayed column generation method as follows:

- Take a *Restricted LP Master Problem*

$$\begin{aligned}\text{(RM)}\quad \tilde{z}^{RM} &= \max \tilde{c}^{\mathrm{T}}\tilde{\lambda}\\ \text{s.t. } \tilde{A}\tilde{\lambda} &= \tilde{b},\\ \tilde{\lambda} &\geq 0,\end{aligned}$$

  obtained by setting all but a few $\lambda_{k,t} = 0$ and forcing them to be non-basic variables. Only the remaining columns of (LPM) need to be generated.

- Using the simplex algorithm, find an optimal solution $\tilde{\lambda}^*$ of (RM) and, by complementary slackness, the corresponding optimal solution $(\pi, \mu) \in \mathbb{R}^m \times \mathbb{R}^K$ of the dual of (RM).

- In parallel, for all $k \in [1, K]$ solve the column generation IPs

$$\begin{aligned}(\text{CGIP}_k)\quad \hat{x}^k &= \arg\max_x\, (c^k - \pi^{\mathrm{T}} A^k)x - \mu_k\\ &\text{s.t. } x \in X^k.\end{aligned}$$

- If the *reduced prices* $\zeta_k := (c^k - \pi^{\mathrm{T}} A^k)\hat{x}^k - \mu_k \leq 0$ for all k, then (π, μ) is (DM)-feasible and $\tilde{\lambda}^*$ is (LPM)-optimal.

- Otherwise, pick k such that $\zeta_k > 0$ and add the column

$$\begin{bmatrix} c^k\hat{x}^k \\ A^k\hat{x}^k \\ e_k \end{bmatrix}$$

  to (RLPM) to obtain a new restricted LP master problem (RLPM$_+$). Re-optimise using warmstarting.

- At every iteration, we monitor our progress toward solving (LPM) by storing the primal and dual bounds

$$\tilde{c}^{\mathrm{T}}\tilde{\lambda}^* \leq z^{LPM} \leq \pi b + \sum_{k=1}^{K} \mu_k + \sum_{k=1}^{K} \zeta_k.$$

11.2 The Branch & Price Algorithm

The application of column generation in the context of branch-and-bound for *binary* IPs is called *branch-and-price*.

- The master problem is in the format, where the vectors $x^{k,t}$ are binary,

$$\begin{aligned} z = \max & \sum_{k=1}^{K}\sum_{t=1}^{T_k}((c^k)^{\mathrm{T}}x^{k,t})\lambda_{k,t} \\ \text{s.t.} & \sum_{k=1}^{K}\sum_{t=1}^{T_k}(A^k x^{k,t})\lambda_{k,t} = b, \\ & \sum_{t=1}^{T_k}\lambda_{k,t} = 1, \quad (k = 1,\dots,K), \\ & \lambda_{k,t} \in \{0,1\}, \quad (t = 1,\dots,T_k;\ k = 1,\dots,K). \end{aligned}$$

- If all subproblems have this same format, we already know how delayed column generation is able to generate dual bounds that can be used in branch & bound.
- We need a branching method that guarantees that that subproblems have the same format as the master problem.
- Since the points $x^{k,t} \in X^k$ are all distinct 0-1 vectors, which are all vertices of the unit hypercube, we have that

$$\tilde{x}^k = \sum_{t=1}^{T_k}\tilde{\lambda}_{k,t}x^{k,t}$$

  is a 0-1 vector if and only if $\tilde{\lambda}$ is integer valued.
- If the optimal solution $\tilde{\lambda}$ of (LPM) (found by delayed column generation) is not integer, there exists therefore κ, j such that $\tilde{x}^{\kappa}_j$, the j-th component of $\tilde{x}^{\kappa}$, is fractional.
- We would like to branch by splitting the feasible set $S = S_0 \cup S_1$ into

$$\begin{aligned} S_0 &= S \cap \{x :\ x^{\kappa}_j = 0\}, \\ S_1 &= S \cap \{x :\ x^{\kappa}_j = 1\}. \end{aligned}$$

  What are the corresponding master problems?

For $\delta = 0, 1$, the requirement that

$$\delta = x^{\kappa}_j = \sum_{t=1}^{T_\kappa}\lambda_{\kappa,t}x^{\kappa,t}_j$$

implies that $x_j^{\kappa,t} = \delta$ for all t with $\lambda_{\kappa,t} > 0$. Therefore, the master problem for S_δ is

$$\begin{aligned} z(S_\delta) = \max & \sum_{k\neq\kappa}\sum_{t=1}^{T_k}((c^k)^{\mathrm{T}}x^{k,t})\lambda_{k,t} + \sum_{t:\, x_j^{\kappa,t}=\delta}((c^\kappa)^{\mathrm{T}}x^{\kappa,t})\lambda_{\kappa,t} \\ \text{s.t. } & \sum_{k\neq\kappa}\sum_{t=1}^{T_k}(A^k x^{k,t})\lambda_{k,t} + \sum_{t:\, x_j^{\kappa,t}=\delta}(A^\kappa x^{\kappa,t})\lambda_{\kappa,t} = b, \\ & \sum_{t=1}^{T_k}\lambda_{k,t} = 1, \quad (k\neq\kappa), \\ & \sum_{t:\, x_j^{\kappa,t}=\delta}\lambda_{\kappa,t} = 1 \\ & \lambda_{k,t}\in\{0,1\}, \quad (t=1,\dots,T_k;\ k=1,\dots,K). \end{aligned}$$

Thus, the problem has the same structure as the master problem for S, but some of the columns are permanently excluded. This has the beneficial effect that the deeper the node in the branch-and-bound tree, the fewer patterns $x^{k,t}$ need to be considered.

The column generation subproblems are unchanged for $k \neq \kappa$,

$$\begin{aligned} \tilde{x}^k = \arg\max\,&((c^k)^{\mathrm{T}} - \pi A^k)x - \mu_k \\ &\text{s.t. } x\in X^k, \end{aligned}$$

but for $k = \kappa$ they take on the new form

$$\begin{aligned} \tilde{x}^\kappa(S_\delta) = \arg\max\,&((c^\kappa)^{\mathrm{T}} - \pi A^\kappa)x - \mu_\kappa \\ &\text{s.t. } x\in X^k, \\ &\quad x_j = \delta. \end{aligned}$$

Similar further restrictions apply of course deeper down the branches, where the subproblems are further branched.

11.3 The Cutting Stock Problem

The above ideas can be applied to solve the cutting-stock problem in an approach developed by Gilmore & Gomoroy.

Example 11.1 (Cutting Stock Problem). *A factory has an unlimited stock of 20-inch paper rolls that it can cut into rolls of smaller widths.*

They receive an order of 301 9-inch paper rolls, 401 8-inch paper rolls, 201 7-inch paper rolls and 501 6-inch paper rolls.

Assuming both trim loss and overproduction are waste, how to fill all the orders under minimal cost?

Details of Branch-and-Price for the Cutting Stock Problem:

- More generally, if the stock rolls have width W and there are m different widths w_i $(i = 1, \dots, m)$ in the order, we can generate all patterns $a_j = [\, a_{1j} \,\cdots\, a_{mj} \,]^{\mathrm{T}}$ of patterns consisting of a_{ij} rolls of width w_i that can be cut into a roll of width W, i.e., such that
$$\sum_{i=1}^{m} w_i a_{ij} \leq W.$$
- Note that $a_{ij} \in \mathbb{Z}_+ = \{0, 1, 2, 3, \dots\}$ for all i, j.
- Conceptually, we assemble the columns a_j into a matrix A (although we never want to generate the full data).
- Decision variables: $x_j \in \mathbb{N}_0$, the number of times pattern j is used $(j = 1, \dots, n)$. (Note: the x_j play the roles of variables $\lambda_{k,t}$ used earlier.)
- Constraints: if there are b_i orders of width w_i, filling the orders requires $Ax \geq b$, and due to the assumption that overproduction is waste, w.l.o.g.,
$$Ax = b.$$
- Objective: minimise $\sum_{j=1}^{n} x_j$, the total number of stock rolls used.
- This yields the IP model (Integer Cutting Stock in form similar to a Master Problem)
$$\text{(ICS)} \quad \begin{aligned} \min \sum_{j=1}^{n} x_j & \\ \text{s.t. } Ax &= b, \\ x &\in \mathbb{Z}_+^n. \end{aligned}$$
- The LP relaxation is given by
$$\text{(LCS)} \quad \begin{aligned} \min \vec{1}^{\mathrm{T}} x & \\ \text{s.t. } Ax &= b, \\ x &\geq 0, \end{aligned}$$
which has the dual
$$\text{(DCS)} \quad \begin{aligned} \max b^{\mathrm{T}} y & \\ \text{s.t. } A^{\mathrm{T}} y &\leq \vec{1}. \end{aligned}$$

- The number n of patterns can be huge, so apply delayed column generation to solve (LCS), e.g., starting with initial restricted pattern set

$$\tilde{A} = \begin{bmatrix} \lfloor W/w_1 \rfloor & & 0 \\ & \ddots & \\ 0 & & \lfloor W/w_m \rfloor \end{bmatrix}$$

- Here we select only m patterns in each simplex iteration, i.e., the columns corresponding to the basic variables. All other variables (the non-basic variables) are forced to zero.
- To solve the restricted subproblem, we only need to solve a linear system,

$$\tilde{x} = \tilde{A}^{-1} b.$$

- Theorem (Complementary Slackness) implies that the optimal dual variables of the restricted LPM are given by $\tilde{y} = \tilde{A}^{-\mathrm{T}} \vec{1}$.
- If $\tilde{y}$ is dual feasible (feasible for (DCS)), $\tilde{x}$ is optimal for (LCS).
- Else there exists a pattern j (a column of the full matrix A that has not yet been generated) corresponding to a non-basic variable x_j such that

$$\sum_{i=1}^{m} a_{ij} \tilde{y}_i > 1.$$

- Although pattern j has not yet been generated, we can find out whether or not it exists by solving the knapsack problem

$$\text{(KS)} \quad p^* = \arg\max_p \tilde{y}^{\mathrm{T}} p$$
$$\text{s.t.} \sum_{i=1}^{m} w_i p_i \leq W,$$
$$p \in \mathbb{Z}_+^m.$$

- If $\tilde{y}^{\mathrm{T}} p^* \leq 1$, then $\tilde{y}$ is dual feasible, and we are in the first case. Else take $(a_{ij})_{i=1}^m := (p_i^*)_{i=1}^m$ as the entering pattern j.
- The exiting variable is determined in the usual simplex fashion, and the dictionary/tableau is pivoted in the usual fashion.

Example 11.2 (CSP continued)**.** *We follow through by applying delayed column generation to solve the LP relaxation of the ICS of our example. Since $\tilde{A}, \tilde{x}, \tilde{y}$ change in each iteration, we write $A^{(k)}, x^{(k)}, y^{(k)}$ for the corresponding data in iteration k, and we write $z^{(k)} = \sum_j x_j^{(k)}$ for the objective value.*

- *Using the initialisation discussed above, we have*

$$A^{(0)} = \begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 3 \end{bmatrix},$$

$$x^{(0)} = (A^{(0)})^{-1} \begin{bmatrix} 301 \\ 401 \\ 201 \\ 501 \end{bmatrix} = \begin{bmatrix} 150.5 \\ 200.5 \\ 100.5 \\ 167 \end{bmatrix},$$

$$z^{(0)} = 618.5, \quad \textit{(objective value)}$$

$$y^{(0)} = \big((A^0)^{-1}\big)^{\mathrm{T}} \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1/2 \\ 1/2 \\ 1/2 \\ 1/3 \end{bmatrix}.$$

- *To find the entering variable x_j, we need to identify whether or not a corresponding column exists and generate it, i.e., solve the knapsack problem*

$$\begin{aligned} \textit{(KP)} \quad p^* = \arg\max_p \ & \frac{1}{2}p_1 + \frac{1}{2}p_2 + \frac{1}{2}p_3 + \frac{1}{3}p_4 \\ \textit{s.t.} \ & 9p_1 + 8p_2 + 7p_3 + 6p_4 \leq 20, \\ & p_i \in \mathbb{Z}_+, \quad (i = 1, \dots, 4). \end{aligned}$$

- *Solving (KP) with branch-and-bound, we find $p^* = [0, 0, 2, 1]^{\mathrm{T}}$. Since $(y^{(0)})^{\mathrm{T}} p^* = 4/3 > 1$, p^* will enter the basic cutting patterns as a column of $A^{(1)}$.*

- *To identify the pattern that corresponds to the leaving basic variable, we must calculate*

$$x^{(0)}./(A^{(0)})^{-1} p^* = [\infty, \infty, 140, 500]^{\mathrm{T}}.$$

It is x_3 that imposes the most restrictive bound, thus the third column of $A^{(0)}$ leaves the basis.

- *Next iteration: For simplicity of solving the restricted LP subproblems we discard the leaving column. This is a variant of delayed column generation that guarantees that the subproblems don't grow in size, but this comes at the expense of possibly having to re-generate a discarded column again at a later iteration.*

$$A^{(1)} = \begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 1 & 3 \end{bmatrix},$$

$$x^{(1)} = (A^{(1)})^{-1} \begin{bmatrix} 301 \\ 401 \\ 201 \\ 501 \end{bmatrix} = \begin{bmatrix} 150.5 \\ 200.5 \\ 100.5 \\ 133.5 \end{bmatrix}, \quad z^{(1)} = 585.$$

Solving another knapsack problem identifies $p^* = [0,1,0,2]^{\mathrm{T}}$ *as entering pattern, and column 4 of* $A^{(1)}$ *as leaving pattern.*

- *Next iteration:*

$$A^{(2)} = \begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 2 & 0 & 1 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 1 & 2 \end{bmatrix}, \quad x^{(2)} = \begin{bmatrix} 150.5 \\ 100.375 \\ 100.5 \\ 200.25 \end{bmatrix}, \quad z^{(2)} = 551.625.$$

- *This time the optimal solution of the knapsack problem yields* $(y^{(2)})^{\mathrm{T}} p^* \leq 1$, *showing that* $y^{(2)}$ *is dual optimal, and hence,* $x^{(2)}$ *is primal optimal.*

- *The dual bound* $z^{(2)} = 551.625$ *could now be used in a branch-and bound algorithm to solve the (ICS).*

- *Alternatively, note that rounding down the usage of each pattern leaves a shortfall of 1 roll of each of the ordered widths, which can be covered by two additional stock rolls cut into patterns* $[1,1,0,0]$ *and* $[0,0,1,1]$. *This shows that*

$$551.625 \leq z_{IP} \leq 550 + 2,$$

and hence, $z_{IP} = 552$ *and we have found the optimal solution.*

- *This rounding procedure cannot be guaranteed to yield the optimal solution in general, but it produces a primal and a dual bound which often sandwich* z_{IP} *in a narrow interval, thus yielding an approximation guarantee.*

12 Lecture 12: Lagrangian Relaxation

12.1 Introductory Example: Uncapacitated Facility Location

Many IPs have a structure

$$\begin{aligned} \text{(IP)} \quad z = \max\, & c^{\mathrm{T}} x \\ \text{s.t. } & Ax \le a \\ & Dx \le d \\ & x \ge 0,\ x \in \mathbb{Z}^n, \end{aligned}$$

such that relaxing the constraints $Dx \le d$ yields a substantially more tractable problem where $Ax \le a$ is a *benign* set of constraints (e.g., totally unimodular) in the sense that the following is easy to solve,

$$\begin{aligned} \max\, & c^{\mathrm{T}} x \\ \text{s.t. } & Ax \le a \\ & x \ge 0, x \in \mathbb{Z}^n. \end{aligned}$$

Thus, we may interpret $Dx \le d$ as "malicous" constraints that render the problem (IP) hard to solve.

Note that what is benign or malicous is in the eye of the beholder, as it may be that

$$\begin{aligned} \max\, & c^{\mathrm{T}} x \\ \text{s.t. } & Dx \le d \\ & x \ge 0, x \in \mathbb{Z}^n \end{aligned}$$

is also an easy problem, but it is really the *combination* of the two constraint sets $Ax \le a$ and $Dx \le d$ that renders the problem hard.

Example 12.1 (Uncapacitated facility location (UFL)). *Consider the uncapacitated facility location problem from Lecture 1,*

$$\begin{aligned} \textit{(IP)} \quad z = \max & \sum_{i \in M} \sum_{j \in N} c_{ij} x_{ij} - \sum_{j \in N} f_j y_j \\ \textit{s.t.} & \sum_{j \in N} x_{ij} = 1 \quad (i \in M) \\ & x_{ij} - y_j \le 0 \quad (i \in M, j \in N) \\ & x \in \mathbb{R}_+^{|M| \times |N|},\ y \in \{0,1\}^{|N|}, \end{aligned}$$

where

- M *is the set of customer locations,*

- *N is the set of potential facility locations,*
- *f_j are the fixed costs for opening facility j,*
- *we replaced the original servicing costs c_{ij} with $-c_{ij}$ to turn the problem into a maximisation problem.*

One may take the viewpoint that it is the demand constraints

$$\sum_{j \in N} x_{ij} = 1, \quad (i \in M) \tag{25}$$

that render the problem hard, because these constraints introduce a functional dependence between the decisions pertaining to different facility locations.

Instead of imposing these constraints, let us add a multiple u_i of each residual

$$1 - \sum_{j \in N} x_{ij}$$

to the objective function. The objective function is now

$$\sum_{i \in M} \sum_{j \in N} c_{ij} x_{ij} - \sum_{j \in N} f_j y_j + \sum_{i \in M} u_i (1 - \sum_{j \in N} x_{ij}),$$

and we say that the constraints (25) have been dualised.

The new problem is called a Lagrangian relaxation,

$$\begin{aligned} \text{(IP}(u)\text{)} \quad z(u) = \max \ & \sum_{i \in M} \sum_{j \in N} (c_{ij} - u_i) x_{ij} - \sum_{j \in N} f_j y_j + \sum_{i \in M} u_i \\ \text{s.t. } & x_{ij} - y_j \leq 0 \quad (i \in M, j \in N) \\ & x \in \mathbb{R}_+^{|M| \times |N|}, \ y \in \{0,1\}^{|N|}. \end{aligned}$$

Note that because the constraints that linked the different facility locations to one another have been subsumed in the objective function, (IP(u)) decouples,

$$z(u) = \sum_{j \in N} z_j(u) + \sum_{i \in M} u_i,$$

where $z_j(u)$ is the optimal solution of the following problem,

$$\begin{aligned} \text{(IP}_j(u)\text{)} \quad z_j(u) = \max \ & \sum_{i \in M} (c_{ij} - u_i) x_{ij} - f_j y_j \\ \text{s.t. } & x_{ij} - y_j \leq 0 \quad (i \in M) \\ & x_{ij} \geq 0 \ (i \in M), \ y_j \in \{0,1\}^{|N|}. \end{aligned}$$

Furthermore, (IP$_j(u)$) is easily solved by inspection:

- *If $y_j = 0$, then $x_{ij} = 0$ for all i, and the objective value is 0.*
- *If $y_j = 1$, then all clients i for which $c_{ij} - u_i > 0$ will be served, and the objective value is $\sum_{i\in M} \max(0, c_{ij} - u_i)$.*

Therefore, $z_j(u) = \max\Big(0, \sum_{i\in M} \max(0, c_{ij} - u_i) - f_j\Big)$.

Definition 12.2 (Relaxation). *A relaxation of an integer programming problem (IP) $z = \max\{f(x) : x \in \mathscr{F}\}$ is any optimisation problem (R)*

$$w = \max\{g(x) : x \in \mathscr{R}\}$$

with feasible set $\mathscr{R} \supseteq \mathscr{F}$ and an objective function $g(x)$ that satisfies $g(x) \geq f(x)$ for all $x \in \mathscr{F}$.

Lemma 12.3 (Dual bounds by relaxation). *If (R) is a relaxation of (IP), then $w \geq z$.*

Proof. See Problem Sheet 4. □

Corollary 12.3.1 (Optimality by relaxation). *Let $x^* \in \arg\max\{g(x) : x \in \mathscr{R}\}$. If $x^* \in \mathscr{F}$ and $g(x^*) = f(x^*)$, then x^* is an optimal solution of (IP).*

Proof. By Lemma (Dual bounds by relaxation) then the following inequality holds for all $x \in \mathscr{F}$,

$$c^{\mathrm{T}} x \leq z \leq w = g(x^*) = c^{\mathrm{T}} x^*.$$

□

Example 12.4 (UFL continued). *Problem (IP(u)) constructed in Example (UFL) is indeed a relaxation:*

- *Giving up on the requirement $\sum_{j\in N} x_{ij} = 1$ constitutes an enlargement of the feasible set.*
- *The restriction of the new objective*

$$\max_{x,y}\ g(x,y) = \sum_{i\in M}\sum_{j\in N} c_{ij}x_{ij} - \sum_{j\in N} f_j y_j + \sum_{i\in M} u_i(1 - \sum_{j\in N} x_{ij})$$

to the feasible set of the UFL coincides with the objective of the latter,

$$\max_{x,y} \sum_{i\in M}\sum_{j\in N} c_{ij}x_{ij} - \sum_{j\in N} f_j y_j,$$

as any (UFL)-feasible solution (x,y) satisfies the demand constraints $\sum_{j\in N} x_{ij} = 1$, which implies

$$\sum_{i\in M} u_i(1 - \sum_{j\in N} x_{ij}) = 0.$$

12.2 Generalisation

Let us now consider an (IP) in the slightly more general form

$$\begin{aligned} \text{(IP)} \quad z = \max\, & c^{\mathrm{T}} x \\ \text{s.t. } & D_1 x \leq d_1, \\ & D_2 x = d_2, \\ & x \in \mathscr{X} = \{x \in \mathbb{R}^n : \, Ax \leq a, \, x \geq 0, \, x \in \mathbb{Z}^n\} \end{aligned}$$

where $\mathscr{X}$ is a feasible set of "benign" type.

We write $D = [D_1^{\mathrm{T}}, D_2^{\mathrm{T}}]^{\mathrm{T}}$ and $d = [d_1^{\mathrm{T}}, d_2^{\mathrm{T}}]^{\mathrm{T}}$ in block form and denote the set of row indices of D that correspond to inequality constraints by $\mathcal{I}$ and indices corresponding to equality constraints by $\mathcal{E}$.

Definition 12.5 (Lagrangian relaxation). *A* Lagrangian relaxation *of (IP) is a problem of the form*

$$\textit{(IP}(u)\textit{)} \quad z(u) = \max\{c^{\mathrm{T}} x + u^{\mathrm{T}}(d - Dx) : \, x \in \mathscr{X}\}$$

where $u \in \mathbb{R}^m$ is a fixed vector Lagrange multipliers *chosen so that $u_i \geq 0$ for $i \in \mathcal{I}$.*

Proposition 12.6 (Lagrangian relaxations). *Problem* $(\mathrm{IP}(u))$ *is a relaxation of problem* (IP).

Proof. The feasible region of (IP(u)) contains that of (IP), since

$$\mathscr{X} \supseteq \mathscr{F} = \{x \in \mathscr{X} : \, D_1 x \leq d_1, \, D_2 x = d_2\}.$$

For all (IP)-feasible x, the objective function of (IP(u)) is at least as large as that of (IP),

$$c^{\mathrm{T}} x + u^{\mathrm{T}}(d - Dx) = c^{\mathrm{T}} x + \sum_{i \in \mathcal{I}} u_i (d_i - D_{i,:} x) \geq c^{\mathrm{T}} x.$$

□

In the context of Lagrangian relaxations, Corollary 12.3.1 can be recast in terms of a complementarity condition:

Proposition 12.7 (Optimality by Lagrangian relaxation). *Let $x(u)$ be an optimal solution of the Lagrangian relaxation*

$$(\mathrm{IP}(u)) \quad x(u) \in \arg\max_x \{c^{\mathrm{T}} x + u^{\mathrm{T}}(d - Dx) : \, x \in \mathscr{X}\}.$$

If $x(u)$ is (IP)-feasible, that is $D_i x \leq d_i$ for all $(i \in \mathcal{I})$ and $D_i x = d_i$ for all $(i \in \mathcal{E})$, and if the complementarity conditions

$$u_i (d_i - [Dx(u)]_i) = 0 \quad \forall \, i \in \mathcal{I}$$

are satisfied, then $x(u)$ is an optimal solution of (IP).

Proof. By complementarity, $z \leq z(u) = c^{\mathrm{T}} x(u) + u^{\mathrm{T}}(d - Dx(u)) = c^{\mathrm{T}} x(u) \leq z$, hence $c^{\mathrm{T}} x(u) = z$. □

12.3 Lagrangian Relaxation of STSP

Example 12.8 (Lagrangian relaxation of STSP). *Recall our formulation of the symmetric travelling salesman problem from Lecture 1,*

$$\begin{aligned}\text{(IP)}\quad z = \min &\sum_{e\in E} c_e x_e\\ \text{s.t.} &\sum_{e\in\delta(i)} x_e = 2 \quad (i\in V)\\ &\sum_{e\in E(S)} x_e \le |S|-1 \quad (S\subset V \text{ s.t. } 2\le |S|\le |V|-1)\\ &x\in\{0,1\}^{|E|},\end{aligned}$$

Lemma 12.9 (Redundant subtour elimination constraints). *Half the subtour elimination constraints $\sum_{e\in E(S)} x_e \le |S|-1$ are redundant.*

Proof. For any x feasible for the LP relaxation of (IP) we have

$$|S| - \sum_{e\in E(S)} x_e = \frac{1}{2}\sum_{i\in S}\sum_{e\in\delta(i)} x_e - \sum_{e\in E(S)} x_e = \frac{1}{2}\sum_{e\in\delta(S,S^c)} x_e,$$

where $\delta(S, S^c)$ is the set of edges in E that are incident to one node from S and one from $S^c := V\setminus S$.

Since $\delta(S,S^c) = \delta(S^c, S)$, we now have

$$|S| - \sum_{e\in E(S)} x_e = \frac{1}{2}\sum_{e\in\delta(S,S^c)} x_e = |S^c| - \sum_{e\in E(S^c)} x_e,$$

and hence, $\sum_{e\in E(S)} x_e \le |S|-1 \Leftrightarrow \sum_{e\in E(S^c)} x_e \le |S^c|-1$. □

Example 12.10 (Lagrangian relaxation of STSP continued).

- *Introduce a new (redundant) constraint $\sum_{e\in E} x_e = n$, obtained by summing all degree constraints.*
- *Eliminate all subtour elimination constraints corresponding to sets S that contain node 1, which are redundant by Lemma (Redundant subtour elimination constraints).*
- *Dualise the degree constraints $\sum_{e\in\delta(i)} x_e = 2$, $(i\ne 1)$.*

This yields the following Lagrangian relaxation of (STSP),

$$\text{(IP}(u)) \quad z(u) = \min \sum_{e=(ij)\in E} (c_e - u_i - u_j)x_e + 2\sum_{i\in V} u_i$$

$$\sum_{e\in\delta(1)} x_e = 2$$

$$\sum_{e\in E(S)} x_e \leq |S| - 1, \quad \forall S \subset V \text{ s.t. } 2 \leq |S| \leq |V| - 1,\ 1 \notin S$$

$$\sum_{e\in E} x_e = n$$

$$x \in \{0,1\}^{|E|}.$$

For notational convenience we included a term $u_1(2-\sum_{e\in\delta(1)} x_e) = 0$ *in the objective.*

Lemma 12.11 (1-Tree Characterisation)**.** *A binary vector* $x \in \{0,1\}^{|E|}$ *is* (IP(u))*-feasible if and only if its support* $E(x) := \{e \in E : x_e = 1\})$ *is a* 1-tree *in* $G = (V, E)$*.*

Proof. $\sum_{e\in\delta(1)} x_e = 2$ guarantees that in the subgraph $G_x := (V, E(x))$ exactly two edges are incident to node 1 .

Constraints $\sum_{e\in E(S)} x_e \leq |S| - 1$ guarantee that when node 1 is removed, then there is no cycle left in $E(x) \setminus \delta(1)$.

$\sum_{e\in E} x_e = n$ guarantees that $|E(x) \setminus \delta(1)| = n - 2$ is a cycle free subgraph on $|V \setminus \{1\}| = n - 1$ nodes, which is only possible if $E(x) \setminus \delta(1)$ is a spanning tree on $V \setminus \{1\}$.

Conversely, if $E(x)$ is a 1-tree, then $\sum_{e\in\delta(1)} x_e = 2$ and $\sum_{e\in E} x_e = n$ are clearly satisfied, and since $E(x)\setminus\delta(1)$ a tree, the subtour elimination constraints are satisfied. □

Example 12.12 (Lagrangian relaxation of STSP continued)**.** *Let us now look at a numerical example and consider the STSP on 5 nodes with edge cost matrix*

$$[c_e] = \begin{bmatrix} - & 30 & 26 & 50 & 40 \\ 30 & - & 24 & 40 & 50 \\ 26 & 24 & - & 24 & 26 \\ 50 & 40 & 24 & - & 30 \\ 40 & 50 & 26 & 30 & - \end{bmatrix}.$$

Note that the Lagrange multipliers u *are unrestricted, as the constraints we dualised were* equality constraints. *Therefore,*

$$u = \begin{bmatrix} 0 & 0 & -15 & 0 & 0 & 0 \end{bmatrix}$$

is a legitimate choice.

Writing $\bar{c}_{ij} := c_{ij} - u_i - u_j$, we obtain the revised edge cost matrix

$$[\bar{c}_e] = \begin{bmatrix} - & 30 & 41 & 50 & 40 \\ 30 & - & 39 & 40 & 50 \\ 41 & 39 & - & 39 & 41 \\ 50 & 40 & 39 & - & 30 \\ 40 & 50 & 41 & 30 & - \end{bmatrix}.$$

Using the greedy algorithm, we find that $\{(1,2),(1,5),(4,5),(2,3),(3,4)\}$ is a minimum weight 1-tree for the revised edge costs. Since this is a Hamiltonian tour, all degree constraints are satisfied and the complementarity condition holds. Proposition (Optimality by Lagrangian relaxation) thus implies that this is an STSP-optimal tour.

12.4 The Lagrangian Dual Problem

In summary so far, it follows from Proposition (Lagrangian relaxations) and Lemma (Dual bounds by relaxation) that for all $u \in \mathbb{R}^m$ with $u_i \geq 0$ for $i \in \mathcal{I}$,

$$z(u) := \max\{c^{\mathrm{T}}x + u^{\mathrm{T}}(d - Dx) : x \in \mathscr{X}\}$$

is a dual bound on the optimal objective value of problem (IP),

$$z = \max\{c^{\mathrm{T}}x : x \in \mathscr{X}, D_1x \leq d_1, D_2x = d_2\}.$$

Definition 12.13 (Lagrangian Dual)**.** *The problem of finding the tightest upper bound obtainable in this fashion can now be cast as an optimisation problem over the Lagrange multipliers u as decision variables,*

$$\textit{(LD)} \quad w_{LD} = \min\{z(u) : u \in \mathbb{R}^m, u_i \geq 0 \ (i \in \mathcal{I})\}.$$

This is called the Lagrangian Dual *of problem (IP),*

Theorem 12.14 (Characterisation of Lagrangian dual bound)**.** *The Lagrangian dual bound is characterised as follows,*

$$\begin{aligned} (\mathrm{LD}') \quad w_{LD} = \max_x \ & c^{\mathrm{T}}x \\ \textit{s.t. } & D_1x \leq d_1 \\ & D_2x = d_2 \\ & x \in \mathrm{conv}(\mathscr{X}). \end{aligned}$$

Proof. We give the proof in the special case where $\mathscr{X} = \{x^{[1]}, \ldots, x^{[T]}\}$ is a finite set. Then

$$\begin{aligned} w_{LD} &= \min_{u_i \geq 0,\, i \in \mathcal{I}} z(u) = \min_{u_i \geq 0,\, i \in \mathcal{I}} \left\{ \max\{c^{\mathrm{T}}x^{[t]} + u^{\mathrm{T}}(d - Dx^{[t]}) : t = 1, \ldots, T\} \right\} \\ &= \min_{(\eta,u) \in \mathbb{R}^{m+1}} \left\{ \eta : \eta \geq c^{\mathrm{T}}x^{[t]} + u^{\mathrm{T}}(d - Dx^{[t]}), \ (t = 1, \ldots, T), \ u_i \geq 0, \ i \in \mathcal{I} \right\} \\ &= \min_{(\eta,u) \in \mathbb{R}^{m+1}} \left\{ \eta + 0^{\mathrm{T}}u : \eta + (Dx^{[t]} - d)^{\mathrm{T}}u \geq c^{\mathrm{T}}x^{[t]}, \ (t = 1, \ldots, T), \ u_i \geq 0, \ i \in \mathcal{I} \right\}. \end{aligned}$$

Taking the dual of the latter LP, strong LP duality implies

$$\begin{aligned}
w_{LD} = \max_{\mu\in\mathbb{R}^T} \Big\{ & \sum_{t=1}^T \mu_t (c^{\mathrm{T}} x^{[t]}) : \sum_{t=1}^T \mu_t (Dx^{[t]} - d)_i \leq 0,\ (i \in \mathcal{I}), \\
& \sum_{t=1}^T \mu_t (Dx^{[t]} - d)_i = 0, \ \ (i \in \mathcal{E}), \quad \sum_{t=1}^T \mu_t = 1,\ \mu \geq 0 \Big\} \\
= \max_{\mu\in\mathbb{R}^T} & \Big\{ c^{\mathrm{T}} x : D_1 x - d_1 \leq 0,\ D_2 x - d_2 = 0,\ x \in \operatorname{conv}(\{x^{[1]}, \dots, x^{[T]}\}) \Big\}.
\end{aligned}$$

□

13 Lecture 13: Lagrangian Dual

13.1 The Strength of the Lagrangian Dual

Let us consider the IP

$$\begin{aligned}
\text{(IP)} \quad z = \max c^{\mathrm{T}} x& \\
\text{s.t. } D_1 x &\leq d_1, \\
D_2 x &= d_2, \\
x &\in \mathscr{X} = \{x \in \mathbb{R}^n : \, Ax \leq a, \, x \geq 0, \, x \in \mathbb{Z}^n\},
\end{aligned}$$

with Lagrangian relaxation

$$\text{(IP}(u)) \quad z(u) = \max\{c^{\mathrm{T}} x + u^{\mathrm{T}}(d - Dx) : \, x \in \mathscr{X}\},$$

and let us compare the dual bounds associated with the Lagrangian Dual and the LP relaxation of (IP),

$$\begin{aligned}
\text{(LD)} \quad w_{LD} &= \min_u \{z(u) : \, u \in \mathbb{R}^m, \, u_i \geq 0 \, \, (i \in \mathcal{I})\}, \\
\text{(LP)} \quad w_{LP} &= \max_x \{c^{\mathrm{T}} x : \, D_1 x \leq d_1, \, D_2 x = d_2, \, Ax \leq a, \, x \geq 0\}.
\end{aligned}$$

Theorem 13.1 (Lagrangian dual and LP relaxation)**.** *The Lagrangian dual bound is at least as tight as the LP relaxation bound,*

$$z \leq w_{LD} \leq w_{LP}.$$

If $\{x \in \mathbb{R}^n : \, Ax \leq a, \, x \geq 0\}$ *is an ideal formulation of* $\mathscr{X}$*, then* $w_{LD} = w_{LP}$*.*

Proof. We have $\mathscr{X} = \{x \in \mathbb{R}^n : \, Ax \leq a, \, x \geq 0\} \cap \mathbb{Z}^n$. Therefore, $\mathscr{X} \subset \{x \in \mathbb{R}^n : \, Ax \leq a, \, x \geq 0\}$, and hence,

$$\operatorname{conv}(\mathscr{X}) \subseteq \operatorname{conv}\left(\{x \in \mathbb{R}^n : \, Ax \leq a, \, x \geq 0\}\right) = \{x \in \mathbb{R}^n : \, Ax \leq a, \, x \geq 0\}.$$

It follows from Theorem (Characterisation of Lagrangian dual) that (LP) is a *relaxation* of (LD), and hence, $w_{\mathrm{LD}} \leq w_{\mathrm{LP}}$, as claimed.

Moreover, if $\{x \in \mathbb{R}^n : \, Ax \leq a, \, x \geq 0\}$ is an ideal formulation of $\mathscr{X}$, then

$$\{x \in \mathbb{R}^n : \, Ax \leq a, \, x \geq 0\} = \operatorname{conv}(\mathscr{X}),$$

so that (LD) and (LP) coincide. □

In the latter situation the Lagrangian Dual offers an alternative to solving the LP relaxation directly in cases where this is too costly.

13.2 Choosing a Lagrangian Dual

Many problems have several reasonable Lagrangian Duals. In this case it is worthwhile thinking about the advantages and disadvantages of the different formulations before starting any calculations.

Example 13.2 (Generalised assignment problem (GAP))**.** *Consider the* generalised assignment problem

$$\begin{aligned}
\textit{(IP)}\quad z = \max \ & \sum_{j=1}^{n}\sum_{i=1}^{m} c_{ij}x_{ij} \\
\textit{s.t.}\ & \sum_{j=1}^{n} x_{ij} \le 1 \quad (i = 1,\dots,m), \\
& \sum_{i=1}^{m} a_{ij}x_{ij} \le b_j \quad (j = 1,\dots,n), \\
& x \in \{0,1\}^{m\times n}.
\end{aligned}$$

In this case we have multiple choices of a Lagrangian dual:

1. Dualising both sets of constraints

$$\begin{aligned}
\textit{(IP}(u)\textit{)}\quad z(u) = \max_x \ & \sum_{j=1}^{n}\sum_{i=1}^{m}(c_{ij} - u_i - a_{ij}v_j)x_{ij} + \sum_{i=1}^{m} u_i + \sum_{j=1}^{n} v_j b_j \\
\textit{s.t.}\ & x \in \{0,1\}^{m\times n}.
\end{aligned}$$

It is certainly easy to solve this IP, as the remaining feasible set $\mathscr{X} = \{0,1\}^{m\times n}$ has the ideal formulation $\{x \in \mathbb{R}^m : 0 \le x_i \le 1\ \forall\, i\}$. By Theorem 2, solving the Lagrangian dual $\min_{u\ge 0} z(u)$ yields the same result as the LP relaxation of (IP). However, it might still be of interest to solve the Lagrangian dual, as (IP(u)) can be solved by inspection:

$$x^*_{ij} = \begin{cases} 1 & \textit{if } c_{ij} - u_i - a_{ij}v_j > 0 \\ 0 & \textit{otherwise.} \end{cases}$$

2. Dualising only the second set of constraints *The advantages and disadvantages of this relaxation are similar to the first case, since the constraint matrix has the consecutive ones property.*

3. Dualising only the first set of constraints

$$\begin{aligned}
\textit{(IP}(u)\textit{)}\quad z(u) = \max_x \ & \sum_{j=1}^{n}\sum_{i=1}^{m}(c_{ij} - u_i)x_{ij} + \sum_{i=1}^{m} u_i \\
\textit{s.t.}\ & \sum_{i=1}^{m} a_{ij}x_{ij} \le b_j \quad (j = 1,\dots,n), \\
& x \in \{0,1\}^{m\times n}.
\end{aligned}$$

Here the remaining feasible set $\mathscr{X} = \{x \in \{0,1\}^{m\times n} : \sum_i a_{ij}x_{ij} \le b_j,\ j = 1,\dots,n\}$ is not of the "easy" type, as

$$\operatorname{conv}(\mathscr{X}) \subset \left\{x \in \mathbb{R}^n : 0 \le x_i \le 1,\ i = 1,\dots,m,\ \sum_i a_{ij}x_{ij} \le b_j\ j = 1,\dots,n\right\}$$

is generally a strict inclusion. Consequently, w_{LD} may be a strictly tighter bound than the bound obtained from the LP relaxation.

The Lagrangian dual is more difficult to solve, but (IP(u)) decouples into blocks $(\{x_{ij} : i = 1,\dots,m\})_{j=1}^n$ and can be parallelised.

13.3 Subgradients

By the definition of a Lagrangian Relaxation, the map

$$u \mapsto z(u) = \max\{c^{\mathrm{T}}x + u^{\mathrm{T}}(d - Dx) : x \in \mathscr{X}\}$$

is a piecewise linear function of u. Therefore, $z(u)$ is a convex function on $\mathscr{D} := \{u \in \mathbb{R}^m : u_i \ge 0,\ (i \in \mathcal{I})\}$ by virtue of the following lemma:

Lemma 13.3 (Pointwise maximum of convex functions)**.** *Let $\{f_i(u) : \mathscr{D} \to \mathbb{R} \big| i \in \mathcal{N}\}$ be a set of convex functions defined on a convex domain $\mathscr{D}$. Then*

$$u \mapsto \max_{i\in\mathcal{N}} f_i(u)$$

is a convex function on $\mathscr{D}$.

Proof. For all $u_1, u_2 \in \mathscr{D}$ and $\lambda \in [0,1]$,

$$\max_i f_i\left(\lambda u_1 + (1-\lambda)u_2\right) \le \max_i\left(\lambda f_i(u_1) + (1-\lambda)f_i(u_2)\right) \le \lambda \max_i f_i(u_1) + (1-\lambda)\max_i f_i(u_2).$$

□

Note that $z(u)$ is not differentiable at breakpoints u where $\arg\max\{c^{\mathrm{T}}x + u^{\mathrm{T}}(d - Dx) : x \in \mathscr{X}\}$ contains more than one point.

Lemma 13.4 (Gradient characterisation of convex functions)**.** *Let $\mathscr{D} \subseteq \mathbb{R}^m$ be a convex domain and $f : \mathscr{D} \to \mathbb{R}$ a convex function with gradient $\gamma = \nabla f(u)$ at $u \in \mathscr{D}$. Then the first order Taylor approximation is a lower bounding function:*

$$f(u) + \gamma^{\mathrm{T}}(v - u) \le f(v), \quad (v \in \mathscr{D}).$$

Proof. By definition, f is convex if for all $u, v \in \mathscr{D}$ and $\lambda \in [0,1]$,

$$f\left(\lambda v + (1-\lambda)u\right) \le \lambda f(v) + (1-\lambda)f(u).$$

Therefore,

$$f(u) + \frac{f\left(u + \lambda(v-u)\right) - f(u)}{\lambda} \le f(v),$$

and taking the limit $\lambda \to 0$ yields the result. □

Definition 13.5 (Extension of a convex function). *Let $\mathscr{D} \subseteq \mathbb{R}^m$ be a convex domain and $f : \mathscr{D} \to \mathbb{R}$ a convex function. We extend f to a* proper convex function *defined on $\mathbb{R}^m$ by setting $f(u) := +\infty$ for $u \notin \mathscr{D}$. The extension satisfies satisfies*

$$f(\lambda u + (1-\lambda)v) \leq \lambda f(u) + (1-\lambda) f(v), \quad (u, v \in \mathbb{R}^m,\ \lambda \in [0,1].$$

Motivated by Lemma (Gradient characterisation of convex functions), the notion of gradient can be generalised to non-differentiable points of convex functions:

Definition 13.6 (Subgradient and Subdifferential). *Let $u \in \mathbb{R}^m$ and let $f : \mathbb{R}^m \to \mathbb{R}$ be a proper convex function. A* subgradient *of f at $u \in \mathbb{R}^m$ is a vector $\gamma \in \mathbb{R}^m$ such that*

$$f(u) + \gamma^{\mathrm{T}}(v-u) \leq f(v), \quad (v \in \mathbb{R}^m).$$

The subdifferential *$\partial f(u)$ of f at u is the set of subgradients of f at u.*

Proposition 13.7 (Properties of the subdifferential). *i) If f is differentiable at u, then $\partial f(u) = \{\nabla f(u)\}$ is a singleton containing only the gradient.*

ii) $\partial f(u)$ is a convex set.

iii) $u^* \in \arg\min f(u)$ if and only if $\vec{0} \in \partial f(u^*)$.

iv) Let $z(u) := \max\{c^{\mathrm{T}}x + u^{\mathrm{T}}(d - Dx) : x \in \mathscr{X}\}$. Then

$$\partial z(u) = \operatorname{conv}\left(\left\{d - Dx^* : x^* \in \arg\max\{c^{\mathrm{T}}x + u^{\mathrm{T}}(d-Dx) : x \in \mathscr{X}\}\right\}\right),$$

where arg max *is the set of all maximisers.*

Proof. See problem sheet. □

13.4 Solving the Lagrangian Dual

Algorithm 13.8 (Subgradient Algorithm for Solving (LD)).

```
Initialise:
    choose $u \in \mathbb{R}^m$ with $u_i \geq 0$, $(i \in \mathcal{I})$;
    $X^* := \arg\max\{c^{\mathrm{T}}x + u^{\mathrm{T}}(d - Dx) : x \in \mathscr{X}\}$,
  $V := \{d - Dx^* : x^* \in X^*\}$;
while $\vec{0} \notin \operatorname{conv}(V)$ do
    choose $v \in V$, $\mu > 0$;
    if $i \in \mathcal{I}$ then
        $u_i := \max(u_i - \mu v_i, 0)$; // Step (*)
    else
        $u_i := u_i - \mu v_i$;
    end
    $X^* := \arg\max\{c^{\mathrm{T}}x + u^{\mathrm{T}}(d - Dx) : x \in \mathscr{X}\}$,
    $V := \{d - Dx^* : x^* \in X^*\}$;
end
```

Notes:

- In each iteration of the main loop the Lagrange multiplier vector is improved by correcting it in a direction $-v$ that makes the objective function $z(u)$ *decrease*.
- Note the built-in safeguard mechanism (Step (*)) that prevents individual components of the updated u to become negative for $i \in \mathcal{I}$.
- The termination criterion of the main loop can be evaluated by solving an LP (see problem sheet).
- The choice of *step length* μ requires further discussion.

We want to choose the step length $\mu > 0$ to guarantee that the algorithm converges to the optimal solution of the Lagrangian Dual. The following lemma gives further insight:

Let

- $u^{[k]}$, $v^{[k]}$ and μ_k be the values of u, v and μ in the k-th iteration of the main loop,
- $z_k := z(u^{[k]})$,
- $U^* := \arg\min_u z(u)$, $w_{\mathrm{LD}} := \min_u z(u)$,
- $\operatorname{dist}(u^{[1]}, U^*) := \min_{u^* \in U^*} \|u^{[1]} - u^*\|_2$ (measures how far from an optimal value we start the iterations).

Lemma 13.9 (Convergence of the subgradient algorithm). *If there exists $G > 0$ such that $\|v^{[k]}\|_2 \le G$ for all k, then*

$$\min_{i \in [1,k]} z_i - z^* \le \frac{\operatorname{dist}(u^{[1]}, U^*) + G^2 \sum_{i=1}^k \mu_i^2}{2 \sum_{i=1}^k \mu_i}.$$

Proof. For any $u^* \in U^*$,

$$\begin{aligned}
\|u^{[k+1]} - u^*\|_2^2 &\le \|u^{[k]} - \mu_k v^{[k]} - u^*\|_2^2 \quad \text{(see problem sheet)} \\
&= \|u^{[k]} - u^*\|_2^2 - 2\mu_k v^{[k]\,\mathrm{T}}(u^{[k]} - u^*) + \mu_k^2 \|v^{[k]}\|_2^2 \\
&\le \|u^{[k]} - u^*\|_2^2 - 2\mu_k (z_k - z^*) + \mu_k^2 G^2 \quad \text{(since } v^{[k]} \text{ is a subgradient).}
\end{aligned}$$

By recursion, $\|u^{[k+1]} - u^*\|_2^2 \le \|u^{[1]} - u^*\|_2^2 - 2\sum_{i=1}^k \mu_i (z_i - z^*) + \sum_{i=1}^k \mu_i^2 G^2$, and hence,

$$\begin{aligned}
2\left(\sum_{i=1}^k \mu_i\right) \times \left(\min_{i \in [1,k]} z_i - z^*\right) &\le 2\sum_{i=1}^k \mu_i (z_i - z^*) + \|u^{[k+1]} - u^*\|_2^2 \\
&\le \|u^{[1]} - u^*\|_2^2 + \sum_{i=1}^k \mu_i^2 G^2.
\end{aligned}$$

□

Corollary 13.9.1 (Basic step length choice)**.** *If* $(\mu_k)_{\mathbb{N}} \subset \mathbb{R}_+$ *is chosen such that*

i) $\sum_{k=1}^{\infty} \mu_k = \infty$,

ii) $\sum_{k=1}^{\infty} \mu_k^2 < \infty$,

then $z(u^{[k]}) \to w_{LD}$.

13.5 Practical Subgradient Algorithm

Theorem 13.10 (Improved step length choice)**.** *i) If* $\sum_k \mu_k \to \infty$ *and* $\mu_k \to 0$ *as* $k \to \infty$, *then* $z(u^{[k]}) \to w_{LD}$.

ii) If $\mu_k = \mu_0 \rho^k$ *for some fixed* $\rho \in (0,1)$ *for* μ_0 *sufficiently large and* ρ *sufficiently close to* 1, *then* $z(u^{[k]}) \to w_{LD}$.

iii) if $z(u^{[0]}) > \overline{w} \geq w_{LD}$ *and*

$$\mu_k = \frac{\varepsilon_k \times \left(z(u^{[k]}) - \overline{w}\right)}{\|v^{[k]}\|^2},$$

where $\varepsilon_k \in (0,2)$ *for all* k, *then either* $z(u^{[k]}) \to w_{LD}$ *for* $k \to \infty$, *or else* $\overline{w} \geq z(u^{[k]}) \geq w_{LD}$ *occurs for some finite* k.

Step length choice iv) gives the most useful step lengths in practice, but note that for μ_k to be positive, we need an upper bound $\overline{w} \in (w_{LD}, z(u^{[k]}))$. In practical applications such a bound is not available explicitly.

Note however:

- Lower bounds $\underline{w}$ of w_{LD} are often available by ways of using heuristics that produce primal feasible solutions.
- If the bound $\overline{w}$ in Rule iv) is chosen too low, μ_k is positive but possibly too large. If $z(u^{[k+1]}) < z(u^{[k]})$, this does not pose a problem, as descent is achieved and the point $u^{[k+1]}$ can be accepted as the next iterate.
- If $z(u^{[k+1]}) \geq z(u^{[k]})$, the step μ_k took the iterate to a point where the objective function $z(u)$ increases again. The guess of $\overline{w}$ then needs to be increased to reduce the step length μ_k.

Algorithm 13.11 (Practical Subgradient Algorithm for Solving (LD))**.**

```
Initialise:
    fix $\varepsilon \in (0,2)$, choose $u \in \mathbb{R}^m$ with $u_i \geq 0$, $(i \in \mathcal{I})$;
    $X^* := \arg\max\{c^{\mathrm{T}}x + u^{\mathrm{T}}(d - Dx) : x \in \mathscr{X}\}$,
$V := \{d - Dx^* : x^* \in X^*\}$;
    find $x \in \mathscr{F} = \mathscr{X} \cap \{D_1 x \leq d_1,\ D_2 x = d_2\}$;
    set $\overline{w} := \underline{w} := c^{\mathrm{T}}x$;
while $\vec{0} \notin \mathrm{conv}(V)$ do
    choose $v \in V$;
    $z^+ := +\infty$;
    while $z^+ \geq z(u)$ do
        $\overline{w} := \frac{z(u)+\overline{w}}{2}$; // this is our guess of a suitable
            dual bound
        $\mu := \frac{\varepsilon(z(u)-\overline{w})}{\|v\|^2}$;
        if $i \in \mathcal{I}$ then
            $u_i^+ := \max\big(u_i - \mu v_i,\ 0\big)$; // compute candidate updates
        else
            $u_i^+ := u_i - \mu v_i$; // compute candidate updates
        end
        $z^+ := z(u^+)$; // evaluate candidate updates
    end
    $u := u^+$; // accept candidate updates as actual
        updates
    $\overline{w} := \underline{w}$;
    $X^* := \arg\max\{c^{\mathrm{T}}x + u^{\mathrm{T}}(d - Dx) : x \in \mathscr{X}\}$,
    $V := \{d - Dx^* : x^* \in X^*\}$;
end
```

13.6 An Example

Example 13.12 (STSP). *We revisit the Lagrangian Dual of the STSP, this time without assuming that we have a priori knowledge of the optimal u.*

The dualised constraints were the degree constraints

$$\sum_{e \in \delta(i)} x_e = 2 \quad (i \in V).$$

Since these are equality constraints*, the Lagrange multipliers u are unconstrained, and the updating rule is*

$$u_i^{[k+1]} = u_i^{[k]} + \mu_k\big(2 - \sum_{e \in \delta(i)} x_e^*(u^{[k]})\big).$$

The STSP being a minimisation *problem rather than a maximisation problem, we have to replace all mins by maxes, lower bounds by upper bounds and so forth.*

We step length rule iii) with $\varepsilon_k = 1$, that is,

$$\mu_k = \frac{\underline{w} - z(u^{[k]})}{\sum_{i\in V}\big(2 - \sum_{e\in\delta(i)} x_e^*(u^{[k]})\big)^2},$$

where $\underline{w}$ is a lower bound *on w_{LD}, and where we had to invert the sign of μ_k because (LD) is a maximisation problem.*

Initialisation: *We apply the greedy heuristic and find the tour $1 \to 2 \to 3 \to 4 \to 5 \to 1$ of length 148.*

As no lower bound $\underline{w}$ on w_{LD} is known at present, we use the primal (upper) bound $\overline{w} = 148$ instead and see how far this allows us to increase $z(u^{[k]})$, knowing that at a later time we will probably have to replace $\overline{w}$ by a smaller value, as 148 is usually not a lower bound.

Iteration 0: *Starting with $u^{[0]} = [0, 0, 0, 0, 0]$, the revised costs are given by*

$$\bar{c}_{ij}^{[0]} = c_{ij} - u_i^{[0]} - u_j^{[0]} = c_{ij}.$$

Solving the associated min-cost 1-tree problem, we find the optimal 1-tree with edge incidence matrix

$$[x_{ij}^*(u^{[0]})] = \begin{bmatrix} - & 1 & 1 & 0 & 0 \\ - & - & 1 & 0 & 0 \\ - & - & - & 1 & 1 \\ - & - & - & - & 0 \\ - & - & - & - & - \end{bmatrix},$$

leading to the objective value $z(u^{[0]}) = 130$.

The subgradient of $z(u)$ at $u^{[0]}$ is

$$\Big[\big(2 - \sum_{e\in\delta(i)} x_e^*(u^{[0]})\big)_{i=1,\dots,m}\Big] = [0, 0, -2, 1, 1],$$

and as $\mu_0 = (148-130)/6 = 3$, we find $u^{[1]} = u^{[0]} + 3\cdot[0, 0, -2, 1, 1] = [0, 0, -6, 3, 3]$.

Iteration 1: *The new cost matrix is*

$$[\bar{c}_{ij}^{[1]}] = \Big[c_{ij} - u_i^{[1]} - u_j^{[1]}\Big] = \begin{bmatrix} - & 30 & 32 & 47 & 37 \\ - & - & 30 & 37 & 47 \\ - & - & - & 27 & 29 \\ - & - & - & - & 24 \\ - & - & - & - & - \end{bmatrix}.$$

The optimal 1-tree is found as

$$[x_{ij}^*(u^{[1]})] = \begin{bmatrix} - & 1 & 1 & 0 & 0 \\ - & - & 1 & 0 & 0 \\ - & - & - & 1 & 0 \\ - & - & - & - & 1 \\ - & - & - & - & - \end{bmatrix},$$

leading to the objective value $z(u^{[1]}) = 143 + 2\sum_i u_i^{[1]} = 143$.

Updating the Lagrange multipliers, we obtain

$$u^{[2]} = u^{[1]} + \frac{148 - 143}{2} \cdot [0, 0, -1, 0, 1] = \left[0, 0, -\frac{17}{2}, 3, \frac{11}{2}\right].$$

Iteration 2: *The new cost matrix and optimal 1-tree are*

$$[\bar{c}_{ij}^{[2]}] = \begin{bmatrix} - & 30 & 34.5 & 47 & 34.5 \\ - & - & 32.5 & 37 & 44.5 \\ - & - & - & 29.5 & 29 \\ - & - & - & - & 21.5 \\ - & - & - & - & - \end{bmatrix}, \qquad [x_{ij}^*(u^{[2]})] = \begin{bmatrix} - & 1 & 0 & 0 & 1 \\ - & - & 1 & 0 & 0 \\ - & - & - & 0 & 1 \\ - & - & - & - & 1 \\ - & - & - & - & - \end{bmatrix},$$

leading to the objective value $z(u^{[2]}) = 147.5$.

This means that 147.5 *is a lower bound on the optimal value* z *of the STSP. But since* $[c_{ij}]$ *are all integer valued, this implies*

$$148 = \lceil 147.5 \rceil \leq z \leq \overline{w} = 148,$$

which shows that the greedy tour $1 \to 2 \to 3 \to 4 \to 5 \to 1$ *was STSP-optimal!*

14 Lecture 14: Cutting Planes

14.1 LP Preprocessing

LP or IP models can often be simplified by reducing the number of variables and constraints, and IP models can be tightened before any actual branch-and-bound computations are performed.

Example 14.1 (Preprocessing an LP). *Consider the LP instance*

$$\begin{aligned} \max\ & 2x_1 + x_2 - x_3 \\ \textit{s.t.}\ & 5x_1 - 2x_2 + 8x_3 \le 15 \\ & 8x_1 + 3x_2 - x_3 \ge 9 \\ & x_1 + x_2 + x_3 \le 6 \\ & 0 \le x_1 \le 3 \\ & 0 \le x_2 \le 1 \\ & 1 \le x_3. \end{aligned}$$

Tightening bounds: *Isolating x_1 in the first constraint and using $x_2 \le 1$, $-x_3 \le -1$ yields*

$$5x_1 \le 15 + 2x_2 - 8x_3 \le 15 + 2 \times 1 - 8 \times 1 = 9,$$

and hence, $x_1 \le 9/5$, which tightens the bound $x_1 \le 3$.

Likewise, isolating x_3 in the first constraint, and using the bound constraints, we find

$$8x_3 \le 15 + 2x_2 - 5x_1 \le 15 + 2 \times 1 - 5 \times 0 = 17.$$

This implies $x_3 \le 17/8$ and tightens $x_3 \le \infty$.

And finally, isolating x_2 in the first constraint,

$$2x_2 \ge 5x_1 + 8x_3 - 15 \ge 5 \times 0 + 8 \times 1 - 15 = -7$$

yields $x_2 \ge -7/2$ which does not tighten $x_2 \ge 0$.

Proceeding similarly with the second and third constraints, we obtain the tightened bound

$$8x_1 \ge 9 - 3x_2 + x_3 \ge 9 - 3 + 1 = 7,$$

yielding the improved bound $x_1 \ge 7/8$.

As some of the bounds have changed after the first sweep, we may now go back to the first constraint and tighten the bounds yet further. Isolating x_3, we obtain

$$8x_3 \le 15 + 2x_2 - 5x_1 \le 15 + 2 - 5 \times \frac{7}{8} = \frac{101}{8},$$

yielding the improved bound $x_3 \le 101/64$.

Continuing the second sweep by isolating each variable in turn in each of the constraints 1–3, and using the bound constraints, several bound constraints may further tighten in general, but not in the present example.

How many sweeps of this process are needed? One can show that after two sweeps of all the constraints and variables, the bounds cannot improve any further!

Redundant Constraints: *Using the final upper bounds in constraint 3,*

$$x_1 + x_2 + x_3 \le \frac{9}{5} + 1 + \frac{101}{64} < 6,$$

so that this constraint is redundant and can be omitted.

The remaining problem is

$$\begin{aligned}\max\ & 2x_1 + x_2 - x_3 \\ & 5x_1 - 2x_2 + 8x_3 \le 15 \\ & 8x_1 + 3x_2 - x_3 \ge 9 \\ & \frac{7}{8} \le x_1 \le \frac{9}{5}, \quad 0 \le x_2 \le 1, \quad 1 \le x_3 \le \frac{101}{64}.\end{aligned}$$

Variable fixing:

- *Increasing x_2 makes the objective function grow and loosens all constraints except $x_2 \le 1$. Therefore, in an optimal solution we must have $x_2 = 1$.*
- *Decreasing x_3 makes the objective function grow and loosens all constraints except $1 \le x_3$. Thus, in an optimal solution we must have $x_3 = 1$.*

This leaves the trivial problem

$$\max\left\{2x_1 : \frac{7}{8} \le x_1 \le \frac{9}{5}\right\}.$$

14.2 IP Preprocessing

In the preprocessing of IPs we have further possibilities:

- For all x_j with an integrality constraint $x_j \in \mathbb{Z}$ any bounds $l_j \le x_j \le u_j$ can be tightened to $\lceil l_j \rceil \le x_j \le \lfloor u_j \rfloor$.
- For binary variables new *logical* or *Boolean* constraints can be derived that tighten the formulation and hence lead to fewer branching nodes in a branch-and-bound procedure.

The latter point is illustrated in the next example:

Example 14.2 (Preprocessing Binary Programming Problems)**.** *Consider a BIP instance whose feasible set is defined by the following constraints,*

$$\begin{aligned} 7x_1 + 3x_2 - 4x_3 - 2x_4 &\leq 1 \\ -2x_1 + 7x_2 + 3x_3 + x_4 &\leq 6 \\ -2x_2 - 3x_3 - 6x_4 &\leq -5 \\ 3x_1 - 2x_3 &\geq -1 \\ x &\in \{0,1\}^4. \end{aligned}$$

Generating logical inequalities: *The first constraint shows that $x_1 = 1 \Rightarrow x_3 = 1$, which can be written as $x_1 \leq x_3$. Likewise, $x_1 = 1 \Rightarrow x_4 = 1$, or equivalently, $x_1 \leq x_4$.*

Finally, constraint 1 also shows that the problem is infeasible if $x_1 = x_2 = 1$. Therefore, the following constraint must hold,

$$x_1 + x_2 \leq 1.$$

We can process the remaining constraints in a similar vein:

- *Constraint 2 yields the inequalities $x_2 \leq x_1$ and $x_2 + x_3 \leq 1$.*
- *Constraint 3 yields $x_2 + x_4 \geq 1$ and $x_3 + x_4 \geq 1$.*
- *Constraint 4 yields $x_1 \geq x_3$.*

Although the introduction of the new logical constraints makes the problem seem more complicated, the formulation becomes tighter and thus easier to solve. Furthermore, we can now process the problem further:

Combining pairs of logical inequalities: *We now consider pairs involving the same variables.*

- *$x_1 \leq x_3$ and $x_1 \geq x_3$ yield $x_1 = x_3$.*
- *$x_1 + x_2 \leq 1$ and $x_2 \leq x_1$ yield $x_2 = 0$, and then $x_2 + x_4 \geq 1$ yields $x_4 = 1$.*

Simplifying: *Substituting the identities $x_2 = 0$, $x_3 = x_1$ and $x_4 = 1$ we found, all four constraints become redundant.*

We are left with the choice $x_1 \in \{0, 1\}$, and hence the feasible set contains only two points

$$S = \{(1, 0, 1, 1), (0, 0, 0, 1)\}.$$

14.3 The Cutting Plane Algorithm

The above discussion of preprocessing steps shows that it is possible to derive valid inequalities for the integer feasible solutions of an IP from its polyhedral formulation. This idea can be generalised and systematically exploited in the design of algorithms.

Consider the problem

$$\begin{aligned}\text{(IP)} \quad & \max\, c^{\mathrm{T}} x \\ & \text{s.t. } x \in \mathscr{X} = \mathscr{P} \cap \mathbb{Z}^n,\end{aligned}$$

where $\mathscr{P} = \{x \in \mathbb{R}^n : a_i^{\mathrm{T}} x = b_i,\ (i = 1, \dots, m),\ x \geq 0\}$.

Definition 14.3 (Valid inequalities and cuts)**.** *A* valid inequality *for $\mathscr{X}$ is an inequality of the form*

$$\alpha^{\mathrm{T}} x \leq \alpha_0$$

that is satisfied for all $x \in \mathscr{X}$ (but not necessarily for all $x \in \mathscr{P}$).

A cut *for $x^* \in \mathscr{P} \setminus \mathscr{X}$ is a valid inequality for $\mathscr{X}$ such that*

$$\alpha^{\mathrm{T}} x^* > \alpha_0,$$

that is, $\mathscr{P} \cap \{x : \alpha^{\mathrm{T}} x \leq \alpha_0\}$ is a tighter formulation of $\mathscr{X}$ that excludes x^*.

Algorithm 14.4 (Cutting Plane Algorithm)**.**

```
solve LP relaxation $x^* = \arg\max_x\{c^{\mathrm{T}} x : x \in \mathscr{P}\}$; // initialisation
while $x^*$ fractional do
    find a cut $\alpha^{\mathrm{T}} x \leq \alpha_0$ for $x^*$;
    $\mathscr{P} \leftarrow \mathscr{P} \cap \{x : \alpha^{\mathrm{T}} x \leq \alpha_0\}$;
    // ($x^*$ is cut from the new formulation)
    solve LP relaxation $x^* = \arg\max_x\{c^{\mathrm{T}} x : x \in \mathscr{P}\}$;
end
```

Notes:

- The algorithm relies on systematic methods to generate cuts, an issue we will discuss further.
- In contrast to the Branch & Bound Algorithm, the convergence of the Cutting Plance Algorithm is not guaranteed but depends on the nature of the cuts that are applied.
- Ideally, one would like to apply cuts that are easily computed and cut off a large part of $\mathscr{P}$, but the two goals are often contradictory.
- The Cutting Plane Algorithm can be combined with the Branch-and-Bound Method, which yields the most powerful black-box solvers for IPs (Branch-and-Cut Algorithm).

14.4 Chvàtal Cuts

Example 14.5 (Chvàtal cut). *Consider the IP* $\min_x\{-x_1 - x_2 - x_3 : x \in \mathscr{X}\}$ *with* $\mathscr{X} = \mathscr{P} \cap \mathbb{Z}^3$ *and*

$$\mathscr{P} = \{x \in \mathbb{R}^3 : x_1 + x_2 \leq 1,\ x_2 + x_3 \leq 1,\ x_1 + x_3 \leq 1,\ x \geq 0\}.$$

Using slack variables x_4, x_5, x_6*, the vector* $x^* = (0.5, 0.5, 0.5, 0, 0, 0)$ *is optimal for the LP relaxation of (IP),*

$$\textit{(LP)} \quad \min_{x\geq 0} -x_1 - x_2 - x_3 \quad \textit{s.t.} \begin{cases} x_1 + x_2 + x_4 = 1 \\ x_2 + x_3 + x_5 = 1 \\ x_1 + x_3 + x_6 = 1 \end{cases}$$

Multiplying the three equality constraints with 0.5 and adding them yields

$$x_1 + x_2 + x_3 + 0.5x_4 + 0.5x_5 + 0.5x_6 = 1.5.$$

Using the non-negativity of x_4, x_5, x_6*, this implies* $x_1+x_2+x_3 \leq 1.5$*, which is a valid inequality not only for* $\mathscr{X}$ *but also for* $\mathscr{P}$*. Now using the integrality of* x_1, x_2, x_3*, we obtain the valid inequality*

$$x_1 + x_2 + x_3 \leq 1,$$

which is a cut for x^* *because* $x_1^* + x_2^* + x_3^* = 1.5$*.*

We now generalise the approach described above: For any $r \in \mathbb{R}^n$, let $\lfloor r \rfloor = [\lfloor r_1 \rfloor, \dots, \lfloor r_n \rfloor]$.

Definition 14.6 (Chvàtal cuts). *Let* $\mathscr{P}^{(0)} = \{x \geq 0 : Ax = b\}$ *be a polyhedron given by a system of* m *equations and* n *non-negativity constraints, and let* $u \in \mathbb{R}^m$*. The Chvàtal cut associated with* u *is given by*

$$\alpha^{\mathrm{T}} x \leq \alpha_0,$$

where $\alpha^{\mathrm{T}} := \lfloor u^{\mathrm{T}} A \rfloor$ *and* $\alpha_0 := \lfloor u^{\mathrm{T}} b \rfloor$*.*

Lemma 14.7 (Chvàtal cuts are valid inequalities). *All Chvàtal cuts are valid inequalities for the set* $\mathscr{X} = \mathscr{P}^{(0)} \cap \mathbb{Z}^n$*.*

Proof. $x \in \mathscr{X}$ implies $Ax = b$, and hence, $u^{\mathrm{T}}Ax = u^{\mathrm{T}}b$. Using $x \geq 0$ this implies $\lfloor u^{\mathrm{T}}A \rfloor x \leq u^{\mathrm{T}}b$, and using integrality of x_i, $(i = 1, \dots, n)$, this implies $\lfloor u^{\mathrm{T}}A \rfloor x \leq \lfloor u^{\mathrm{T}}b \rfloor$. □

We state the next result without proof.

Theorem 14.8 (Separation of non-integral vertices). *Let* x^* *be a vertex of* $\mathscr{P}^{(0)}$ *such that* $x^* \notin \mathbb{Z}^n$*. Then there exists a vector* $u \in \mathbb{R}^m$ *such that*

$$\lfloor u^{\mathrm{T}}A \rfloor x \leq \lfloor u^{\mathrm{T}}b \rfloor$$

is a cut for x^**.*

14.5 Chvàtal Closures

In general, not all valid inequalities for $\mathscr{X}$ can be obtained as Chvàtal cuts of a given formulation $\mathscr{P}^{(0)}$:

Let us write $A^{(1)}x \leq b^{(1)}$ for the set of all Chvàtal cuts that can be derived from the formulation $\mathscr{P}^{(0)}$ of $\mathscr{X}$, and let us define the new set

$$\mathscr{P}^{(1)} := \{x \geq 0 :\ Ax = b,\ A^{(1)}x \leq b^{(1)}\}.$$

The next result is given without proof.

Lemma 14.9 (Polyhedrality of Chvàtal closure). *Only finitely many of the inequalities $A^{(1)}x \leq b^{(1)}$ are essential (non-redundant), that is, $\mathscr{P}^{(1)}$ is a polyhedron, and we assume without loss of generality that the system $A^{(1)}x \leq b^{(1)}$ lists only essential Chvàtal cuts, and that there are m_1 of these.*

We can now repeat this construction an apply new Chvàtal cuts

$$\lfloor u^{\mathrm{T}}A + w_1^{\mathrm{T}}A^{(1)} \rfloor x \leq \lfloor u^{\mathrm{T}}b + w_1^{\mathrm{T}}b^{(1)} \rfloor$$

with $u \in \mathbb{R}^m$ and $w_1 \in \mathbb{R}_+^{m_1}$ to $\mathscr{P}^{(1)}$. Denote the essential inequalities of this form by $A^{(2)}x \leq b^{(2)}$. By Lemma (Polyhedrality of Chvàtal closure), there are a finite number m_2 of new inequalities, hence

$$\mathscr{P}^{(2)} := \{x \geq 0 :\ Ax = b,\ A^{(1)}x \leq b^{(1)},\ A^{(2)}x \leq b^{(2)}\}$$

is a polyhedron.

Applying this process iteratively, for $k \in \mathbb{N}$ we obtain polyhedra

$$\mathscr{P}^{(k)} = \{x \geq 0 :\ Ax = b,\ A^{(1)}x \leq b^{(1)}, \dots,\ A^{(k-1)}x \leq b^{(k-1)}\}$$

that generate a new set of m_k inequalities $A^{(k)}x \leq b^{(k)}$ defined as the set of essential Chvàtal cuts of the form

$$\lfloor u^{\mathrm{T}}A + \sum_{j=1}^{k} w_j^{\mathrm{T}}A^{(j)} \rfloor x \leq \lfloor u^{\mathrm{T}}b + \sum_{j=1}^{k} w_j^{\mathrm{T}}b^{(j)} \rfloor$$

with $u \in \mathbb{R}^m$ and $w_j \in \mathbb{R}_+^{m_j}$, $(j = 1, \dots, k)$.

It is clear from the above construction that

$$\mathscr{P}^{(0)} \supseteq \mathscr{P}^{(1)} \supseteq \cdots \supseteq \mathscr{P}^{(k)} \supset \operatorname{conv}(\mathscr{X})$$

is a nesting of ever tighter formulations of $\mathscr{X}$. The next result, given without proof, shows that this process is finite.

Theorem 14.10 (Finiteness of Chvàtal rank). *For any polyhedron $\mathscr{P}^{(0)}$ and $\mathscr{X} = \mathscr{P}^{(0)} \cap \mathbb{Z}^n$ there exists a finite k for which $\mathscr{P}^{(k)} = \operatorname{conv}(\mathscr{X})$.*

Definition 14.11 (Chvàtal closures)**.** *$\mathscr{P}^{(k)}$ is called the k-th* Chvàtal closure *of $\mathscr{P}^{(0)}$, and $\min\{k : \mathscr{P}^{(k)} = \mathrm{conv}(\mathscr{X})\}$ is called the* Chvàtal rank *of $\mathscr{P}^{(0)}$. In particular, $\mathscr{P}^{(0)}$ is an ideal formulation if and only if its Chvàtal rank is zero.*

Notes:

- We started with an equality constrained IP $\max\{c^{\mathrm{T}}x : Ax = b\}$, which is why the vectors u are unconstrained, but the vectors w_k used to take linear combinations of added inequalities $A^{(k)}x \leq b^{(k)}$ must be chosen $w_1 \geq 0$. Alternatively, note that $A^{(k)}, b^{(k)}$ are integer valued, so that if the IP is brought into equality constrained form by adding slack variables $s^{(k)} = b^{(k)} - A^{(k)}x$, these are forced to be integer valued variables too.
- The use of an essential Chvàtal cut in each iteration of the cutting plane algorithm guarantees finite termination, because an ideal formulation of $\mathscr{X}$ is found after applying finitely many iterations. However, there exist examples for which the Chvàtal rank is larger than the number n of decision variables.
- In general even the number of essential inequalities $A^{(1)}x \leq b^{(1)}$ in the first Chvàtal closure is of order $m_1 = O\Big(\binom{n}{m}\Big)$. Applying the cutting plane algorithm with Chvàtal cuts is thus very inefficient in practice.
- However, to solve an IP it is not necessary to find an ideal formulation. A more economical approach is required that only applies cuts that make the optimal solution emerge.

15 Lecture 15: Gomoroy Cuts

15.1 Gomoroy Cuts

Let $\mathscr{P} = \{x \in \mathbb{R}^n : a_i^{\mathrm{T}} x = b_i, \ (i = 1, \dots, m), \, x \geq 0\}$ and consider the IP problem

$$\text{(IP)} \quad \max c^{\mathrm{T}} x \qquad \text{s.t. } x \in \mathscr{X} = \mathscr{P} \cap \mathbb{Z}^n.$$

The LP relaxation is given by

$$\text{(LP)} \quad \max\{c^{\mathrm{T}} x : Ax = b, \, x \geq 0\}$$

with $A = [\, a_1 \, \cdots \, a_m \,]^{\mathrm{T}}$ and $b = [\, b_1 \, \cdots \, b_m \,]^{\mathrm{T}}$. Let x^* be obtained by solving (LP) with the simplex algorithm.

By permuting decision variables we may assume w.l.o.g. that $B = [1, m]$, i.e., the columns so that the basic variables associated with x^* appear in the first $m \times m$ block. The optimal tableau looks as follows,

$$\begin{array}{cc|c} \mathrm{I}_m & \overline{A}_N & \bar{b} \\ \hline 0 & \bar{c}_N^{\mathrm{T}} & -z^* \end{array}$$

where $z^* = c^{\mathrm{T}} x^*$, $\overline{A} = A_B^{-1} A$, $\bar{b} = A_B^{-1} b = x_B^*$ and $\bar{c}_N = c_N - A_N^{\mathrm{T}} A_B^{-\mathrm{T}} c_B$ (see Lecture 2).

If $\bar{b} \in \mathbb{Z}^n$, x^* solves (IP). Otherwise there exists $t \in B$ s.t. x_t^* is fractional and the t-th row of the tableau reads

$$x_t + \sum_{j \in N} \bar{a}_{tj} x_j = \bar{b}_t,$$

Definition 15.1 (Gomoroy Cut)**.** *The* Gomoroy cut *associated with variable x_t^* is the valid inequality*

$$x_t + \sum_{j \in N} \lfloor \bar{a}_{tj} \rfloor x_j \leq \lfloor \bar{b}_t \rfloor.$$

Lemma 15.2. *The Gomoroy cut associated with variable x_t^* is a cut for x^*. This is a special case of a Chvàtal cut.*

Proof. Let e_t be the t-th canonical unit vector. The Gomoroy cut is then obtained as the Chvàtal cut associated with vector $u = A_B^{-T} e_t$, and it is a cut for x^* since

$$x_t^* + \sum_{j \in N} \lfloor \bar{a}_{tj} \rfloor x_j^* = x_t^* > \lfloor \bar{b}_t \rfloor.$$

Indeed, we have

$$u^{\mathrm{T}}A = e_t^{\mathrm{T}} A_B^{-1} A = e_t^{\mathrm{T}} \overline{A} = \overline{a}_t^{\mathrm{T}},$$
$$u^{\mathrm{T}}b = e_t^{\mathrm{T}} A_B^{-1} b = e_t^{\mathrm{T}} \overline{b} = \overline{b}_t^{\mathrm{T}}.$$

□

Subtracting the equality constraint $x_t + \sum_{j\in N} \overline{a}_{tj} x_j = \overline{b}_t$ from its associated Gomoroy cut

$$x_t + \sum_{j\in N} \lfloor \bar{a}_{tj} \rfloor x_j \le \lfloor \bar{b}_t \rfloor,$$

we obtain $\sum_{j\in N} -\varphi(\bar{a}_{tj}) x_j \le -\varphi(\bar{b}_t)$, where we write $\varphi(a) = a - \lfloor a \rfloor$ for the fractional part of any real number a. This yields the Gomoroy cut in *fractional form*,

$$\sum_{j\in N} -\varphi(\bar{a}_{tj}) x_j + s = -\varphi(\bar{b}_t), \quad s \in \mathbb{N}_0,$$

where s is a nonnegative *integer valued* slack variable.

15.2 Cutting Planes with Gomoroy Cuts

We may now use Gomoroy cuts as a way to generate cutting planes in the cutting plane algorithm. Adding the cut to the formulation of our IP, we have

$$\begin{aligned} \text{(IP)} \quad & \max \left[\begin{smallmatrix} c \\ 0 \end{smallmatrix}\right]^{\mathrm{T}} \left[\begin{smallmatrix} x \\ s \end{smallmatrix}\right] \\ & \text{s.t. } \left[\begin{smallmatrix} x \\ s \end{smallmatrix}\right] \in \mathscr{X}^+ := \mathscr{P}^+ \cap \mathbb{Z}^{n+1}, \end{aligned}$$

where

$$\mathscr{P}^+ = \{\left[\begin{smallmatrix} x \\ s \end{smallmatrix}\right] : x \in \mathscr{P}\} \cap \left\{ \left[\begin{smallmatrix} x \\ s \end{smallmatrix}\right] : \sum_{j\in N} -\varphi(\bar{a}_{tj}) x_j + s = -\varphi(\bar{b}_t) \right\}.$$

Writing $\varphi(\bar{a}_{t,:})$ for the vector $[\varphi(\bar{a}_{t,j})]_{j=1}^n$, the LP relaxation of this new formulation is given by

$$\begin{aligned} \text{(LP')} \quad & \max_{x,s} c^{\mathrm{T}} x \\ & \text{s.t. } \begin{bmatrix} \overline{A} & 0 \\ -\varphi(\bar{a}_{t,:}) & 1 \end{bmatrix} \begin{bmatrix} x \\ s \end{bmatrix} = \begin{bmatrix} \bar{b} \\ -\varphi(\bar{b}_t) \end{bmatrix}, \\ & x, s \ge 0. \end{aligned}$$

The Gomoroy cut can be added to the tableau corresponding to x^*, which now has one extra row (for the cut) and column (for the new basic variable s),

$$\begin{array}{ccc|c} \mathrm{I}_m & \overline{A}_N & 0 & \bar{b} \\ 0 & -\varphi(\bar{a}_{t,N}) & 1 & -\varphi(\bar{b}_t) \\ \hline 0 & \bar{c}_N^{\mathrm{T}} & 0 & -z^* \end{array}$$

which now represents a basic (but infeasible) solution of (LP').

However, we can re-optimise the primal tableau via dual simplex steps, because the dual solution y^* remains feasible by extending it to $(y^*, 0)$.

15.3 An Example

Example 15.3 (IP by cutting plane algorithm with Gomoroy cuts)**.**

$$\begin{aligned} \textit{(IP)} \quad & \max_x \, x_2 \\ & \textit{s.t.} \; 3x_1 + 2x_2 + x_3 = 6 \\ & \quad -3x_1 + 2x_2 + x_4 = 0 \\ & \quad x_j \geq 0, \, x_j \in \mathbb{Z}, \; (j = 1, \dots, 4). \end{aligned}$$

The initial tableau is found as

$$\begin{array}{cccc|c} 3 & 2 & 1 & 0 & 6 \\ -3 & 2 & 0 & 1 & 0 \\ \hline 0 & 1 & 0 & 0 & 0 \end{array}$$

and after a few pivoting steps, the optimal tableau is found:

$$\begin{array}{cccc|c} 1 & 0 & 1/6 & -1/6 & 1 \\ 0 & 1 & 1/4 & 1/4 & 3/2 \\ \hline 0 & 0 & -1/4 & -1/4 & -3/2 \end{array}$$

from which we read off the LP-optimal solution $x^* = [1, 3/2, 0, 0]$. *Since* x_2^* *is fractional, row* 2 *yields*

$$\begin{aligned} x_2 + 0.25x_3 + 0.25x_4 &= 1.5, \\ \Rightarrow x_2 + 0 \times x_3 + 0 \times x_4 &\leq \lfloor 1.5 \rfloor, \\ \Rightarrow x_2 &\leq 1. \end{aligned}$$

Introduction of a slack variable x_5 *and subtraction of the two equations yields*

$$\begin{aligned} x_2 + x_5 = 1, \quad & x_5 \geq 0, \, x_5 \in \mathbb{Z} \\ -(x_2 + 0.25x_3 + 0.25x_4 = 1.5) & \\ \Rightarrow -0.25x_3 - 0.25x_4 + x_5 = -0.5, \quad & x_5 \geq 0, \, x_5 \in \mathbb{Z}. \end{aligned}$$

Add this equation to the tableau,

$$\begin{array}{ccccc|c}
1 & 0 & 1/6 & -1/6 & 0 & 1 \\
0 & 1 & 1/4 & 1/4 & 0 & 3/2 \\
0 & 0 & -1/4 & -1/4 & 1 & -1/2 \\
\hline
0 & 0 & -1/4 & -1/4 & 0 & -3/2
\end{array}$$

This tableau describes a basic solution (the intersection of m active constraints) by setting $x_N = 0$, but the solution is not feasible, because $x_5 = -1/2$. To render this variable non-negative, we must use row 3 as a pivot row in which to eliminate a different variable.

To decide on which column to pivot, note that if the pivot row t were to read

$$\sum_{j=1}^{n} \bar{a}_{tj} x_j = \bar{b}_t$$

with $\bar{a}_{tj} \geq 0$ $(j = 1, \dots, n)$ and $\bar{b}_t < 0$, then no matter how $x \geq 0$ is chosen, constraint t cannot be satisfied. In that case, we would have to conclude that the primal problem is infeasible.

Luckily, in our case, this is not so, and in pivoting on column h with $\bar{a}_{th} < 0$, the last row changes as follows,

$$\bar{c}_j \leftarrow \bar{c}_j - \frac{\bar{c}_h}{\bar{a}_{th}} \bar{a}_{tj}.$$

To ensure that dual feasibility is not destroyed, we must not allow any $\bar{c}_j$ to become positive, that is,

$$\begin{aligned}
&\bar{c}_j - \frac{\bar{c}_h}{\bar{a}_{th}} \bar{a}_{tj} \leq 0, \quad (j = 1, \dots, n) \quad \textit{(not a problem if } \bar{a}_{tj} \geq 0\textit{, since } \bar{c}_h \leq 0 \textit{ and } \bar{a}_{th} < 0\textit{)} \\
\Leftrightarrow\ &\frac{\bar{c}_j}{|\bar{a}_{tj}|} - \frac{\bar{c}_h}{|\bar{a}_{th}|} \leq 0, \quad (j \in [1, n],\ \bar{a}_{tj} < 0) \\
&\Leftrightarrow h \in \arg\max \left\{ \frac{\bar{c}_j}{|\bar{a}_{tj}|} : j \in [1, n],\ \bar{a}_{tj} < 0 \right\}.
\end{aligned}$$

For example in our case, $t = 3$, $h \in \{3, 4\}$. Eliminating x_3 in row 3 reoptimises the tableau,

$$\begin{array}{ccccc|c}
1 & 0 & 1/6 & -1/6 & 0 & 1 \\
0 & 1 & 1/4 & 1/4 & 0 & 3/2 \\
0 & 0 & 1 & 1 & -4 & 2 \\
\hline
0 & 0 & -1/4 & -1/4 & 0 & 3/2
\end{array}
\ \rightarrow\
\begin{array}{ccccc|c}
1 & 0 & 0 & -1/3 & 2/3 & 2/3 \\
0 & 1 & 0 & 0 & 1 & 1 \\
0 & 0 & 1 & 1 & -4 & 2 \\
\hline
0 & 0 & 0 & 0 & -1 & -1
\end{array}$$

Row 1 yields the Gomoroy cut $x_1 - x_4 \leq 0$, or in fractional form,

$$\begin{aligned}
x_1 - x_4 + x_6 = 0, &\quad x_6 \geq 0,\ x_6 \in \mathbb{Z} \\
-(x_1 - 1/3x_4 + 2/3x_5 = 2/3)& \\
\Rightarrow -2/3x_4 - 2/3x_5 + x_6 = -2/3, &\quad x_6 \geq 0,\ x_6 \in \mathbb{Z}.
\end{aligned}$$

Add this equation to the tableau,

$$\begin{array}{cccccc|c}
1 & 0 & 0 & -1/3 & 2/3 & 0 & 2/3 \\
0 & 1 & 0 & 0 & 1 & 0 & 1 \\
0 & 0 & 1 & 1 & -4 & 0 & 2 \\
0 & 0 & 0 & -2/3 & -2/3 & 1 & -2/3 \\
\hline
0 & 0 & 0 & 0 & -1 & 0 & -1
\end{array}$$

Now $t = 4$, $h = 4$. Eliminating x_4 in row 4 reoptimises the tableau,

$$\begin{array}{cccccc|c}
1 & 0 & 0 & -1/3 & 2/3 & 0 & 2/3 \\
0 & 1 & 0 & 0 & 1 & 0 & 1 \\
0 & 0 & 1 & 1 & -4 & 0 & 2 \\
0 & 0 & 0 & 1 & 1 & -3/2 & 1 \\
\hline
0 & 0 & 0 & 0 & -1 & 0 & -1
\end{array}
\rightarrow
\begin{array}{cccccc|c}
1 & 0 & 0 & 0 & 1 & -1/2 & 1 \\
0 & 1 & 0 & 0 & 1 & 0 & 1 \\
0 & 0 & 1 & 0 & -5 & 3/2 & 1 \\
0 & 0 & 0 & 1 & 1 & -3/2 & 1 \\
\hline
0 & 0 & 0 & 0 & -1 & 0 & -1
\end{array}$$

The new optimal solution is $x^* = [1, 1, 1, 1, 0, 0]$, and the IP is now solved by LP relaxation.

Note: Alternatively, the second Gomoroy cut $x_1 - x_4 \leq 0$ could have also been reformulated using the substitution $x_4 = 3x_1 - 2x_2$, yielding

$$x_2 \leq x_1.$$

Notes:

- The use of Gomoroy cuts is easy to understand and apply, and it guarantees that the cutting plane algorithm converges. However, the technique is not particularly effective, because the cuts become very shallow very quickly.
- To help the method select deeper cuts, it is advised to generate the Gomoroy cut from the *most fractional* variable x_j^*, that is from the row corresponding to $\arg\max_j \varphi(x_j^*)$.
- If the objective vector c is integer, then the objective z associated with any integer feasible solution is integer, and one can also use the last row

$$-z + \sum_{j \in N} \bar{c}_j x_j = \bar{c}_0$$

  of the optimised tableau to generate a Gomoroy cut when $\bar{c}_0$ is fractional:

$$\sum_{j \in N} \lfloor \bar{c}_j \rfloor x_j \leq \lfloor \bar{c}_0 \rfloor + z$$

$$\sum_{j \in N} \bar{c}_j x_j = \bar{c}_0 + z,$$

  which yields

$$\sum_{j \in N} -\varphi(\bar{c}_j) x_j \leq -\varphi(\bar{c}_0).$$

This is a cut for x^*, since $\sum_{j \in N} \varphi(\bar{c}_j) x_j^* = 0$, but $\varphi(\bar{c}_0) > 0$, as $\bar{c}_0$ was assumed fractional.

16 Lecture 16: Branch and Cut

16.1 The Branch & Cut Framework

Data Specification 16.1 (Branch & Cut)**.**

```
// input data
objective $\max c^{\mathrm{T}}x$;
constraints $Ax = b$ with $A, b$ integer valued, defining feasible set;
$\mathscr{X} = \{x \in \mathbb{Z}^n : Ax = b,\ x \geq 0\}$;
// global data
global pool of valid inequalities $A^{(0)}x \leq b^{(0)}$ for $\mathscr{X}$;
global dual bound $\bar{z}$;
global primal bound $\underline{z}$;
list AN of active nodes;
node counter $k$;
// local (node level) data
local inequalities $A^{[j]}x \leq b^{[j]}$;
local polyhedron $\mathscr{P}^{[j]}$, defining node feasible set $\mathscr{X}^{[j]} = \mathscr{P}^{[j]} \cap \mathbb{Z}^n$;
node dual bound $\bar{z}^{[j]}$;
node primal bound $\underline{z}^{[j]}$;
```

Methods 16.2 (Branch & Cut)**.**

```
function [$x^{[j]}$, $\mathscr{P}^{[j]}$] = addCuts($x^{[j]}$, $\mathscr{P}^{[j]}$)
// input $x^{[j]} = \arg\max\{c^{\mathrm{T}}x : x \in \mathscr{P}^{[j]}\}$ optimised via simplex
while $\mathscr{P}^{[j]} \neq \emptyset$, $x^{[j]}$ fractional and $\exists$ sufficiently deep cut in pool do
    find cut $A_i^{[0]}x^{[j]} > b_i^{[0]}$ in pool of valid inequalities;
    $\mathscr{P}^{[j]} \leftarrow \mathscr{P}^{[j]} \cap \{x : A_i^{[0]}x \leq b_i^{[0]}\}$;
    re-optimise $x^{[j]}$ via dual simplex;
end
while $\mathscr{P}^{[j]} \neq \emptyset$, $x^{[j]}$ fractional and sufficiently deep new cuts generated do
    generate cut $\alpha^{\mathrm{T}}x^{[j]} > \alpha_0$ with $\alpha^{\mathrm{T}}x \leq \alpha_0$ a valid inequality for $\mathscr{X}$;
    $\{A^{[0]}x \leq b^{[0]}\} \leftarrow \{A^{[0]}x \leq b^{[0]}\} \cup \{\alpha^{\mathrm{T}}x \leq \alpha_0\}$; // add valid
        inequality to pool
    $\mathscr{P}^{[j]} \leftarrow \mathscr{P}^{[j]} \cap \{x : \alpha^{\mathrm{T}}x \leq \alpha_0\}$;
    re-optimise $x^{[j]}$ via dual simplex;
end
if $\mathscr{P}^{[j]} = \emptyset$ then
    $x^{[j]} = NaN$;
end
```

Algorithm 16.3 (Branch & Cut).

```
$k = 1$, $A^{[1]}, b^{[1]} = \emptyset$, $AN = \{1\}$, $\underline{z} = -\infty$, $\bar{z} = +\infty$, $x^* = NaN$,
    $\{A^{[0]}x \le b^{[0]}\} = \emptyset$; // initialisation
while $AN \neq \emptyset$ do
    choose $j \in AN$;
    if $\bar{z}^{[j]} \le \underline{z}$ then
        $AN \leftarrow AN \setminus \{j\}$; // pruning
    else
        $\mathscr{P}^{[j]} = \{x \in \mathbb{R}^n : Ax = b,\ A^{[j]}x \le b^{[j]},\ x \ge 0\}$;
        $x^{[j]} := \arg\max\{c^{\mathrm{T}}x : x \in \mathscr{P}^{[j]}\}$; // simplex warmstart
            from parent's initial subproblem
        $[x^{[j]}, \mathscr{P}^{[j]}] = addCuts(x^{[j]}, \mathscr{P}^{[j]})$; // global pool of cuts
            mined and expanded here
        $\bar{z}^{[j]} := c^{\mathrm{T}}x^{[j]}$; // $\bar{z}^{[j]} := -\infty$ if $x^{[j]} = \mathrm{NaN}$
        $\bar{z} := \max\{\bar{z}^{[\ell]} : \ell \in AN\}$; // update global upper bound
        if $x^{[j]}$ integer valued then $y^{[j]} := x^{[j]}$;
        else attempt to find $y^{[j]} \in \mathscr{X}^{[j]}$ via heuristic;
        $\underline{z}^{[j]} := c^{\mathrm{T}}y^{[j]}$; // set $\underline{z}^{[j]} := -\infty$ if $y^{[j]}$ unassigned
        if $\underline{z}^{[j]} > \underline{z}$ then
            $x^* := y^{[j]}$; // update incumbent
            $\underline{z} := \underline{z}^{[j]}$; // update global lower bound
        end
        if $\exists x_\ell^{[j]}$ fractional then
            $\{A^{[k+i]}x \le b^{[k+i]}\} := \{A^{[j]}x \le$
                $b^{[j]}\} \cup \begin{cases} \{x \le \lfloor x_\ell^{[j]} \rfloor\}, & (i = 1), \\ \{x \ge \lceil x_\ell^{[j]} \rceil\}, & (i = 2) \end{cases}$ // cuts from
                global pool not passed on to child nodes!
            $\bar{z}^{[k+i]} := \bar{z}^{[j]}, (i = 1, 2)$; // inherit dual bound from
                parent
            $AN \leftarrow (AN \setminus \{j\}) \cup \{k+1, k+2\}$, $k \leftarrow k + 2$; // branching
        end
    end
end
```

16.2 Algorithm Design Steps

Adapting the branch-and-cut framework to a specific problem class involves the following steps:

1. Identification of structural properties of the problem class.
2. Use polyhedral analysis to translate the structural properties into classes of valid inequalities to be used as cuts.

3. For each class C of valid inequalities, identify an efficient procedure to solve the associated separation problem: Given $x^* \notin \mathscr{X}$, find a valid inequality $\alpha^{\mathrm{T}} x \leq \alpha_0$ for $\mathscr{X}$ among class C such that $\alpha^{\mathrm{T}} x^* > \alpha_0$.

Example 16.4 (Search index construction)**.** *The following example is due to Fischetti:*

Relational data bases use a small number of search indices to use in queries of m different types. We need to choose which among a set of candidate indices $\{1, \dots, n\}$ to build and maintain so as to minimise the expected cost. $j = 0$ represents a default index available without fixed cost:

$$\begin{aligned}
\min_{y,x} \; & \sum_{j=1}^{n} c_j y_j + \sum_{i=1}^{m} \sum_{j=0}^{n} \gamma_{ij} x_{ij} \\
\text{s.t.} \; & \sum_{j=1}^{n} d_j y_j \leq D \\
& \sum_{j=0}^{n} x_{ij} = 1, \quad (i = 1, \dots, m) \\
& \sum_{i=1}^{m} x_{ij} \leq m y_j, \quad (j = 1, \dots, n) \\
& x_{ij},\, y_j \in \{0, 1\}, \quad (i = 1, \dots, m;\, j = 0, \dots, n).
\end{aligned}$$

- $x_{ij} = 1$ *iff using index j for type i query,*
- d_j *memory requirement for index j,*
- *D memory budget for all indices,*
- $x_{ij} = 1$ *for some $i \Rightarrow y_j = 1$ expressed as big M contraint,*
- *index 0 free and available.*

In our example, let $n = 5$, $m = 6$, $D = 19$,

$$[\gamma_{ij}] = \begin{bmatrix} 6200 & 1300 & 6200 & 6200 & 6200 & 6200 \\ 2000 & 900 & 700 & 2000 & 2000 & 2000 \\ 800 & 800 & 800 & 800 & 800 & 800 \\ 6700 & 6700 & 6700 & 1700 & 6700 & 2700 \\ 5000 & 5000 & 5000 & 2200 & 1200 & 4200 \\ 2000 & 2000 & 2000 & 2000 & 2000 & 750 \end{bmatrix}$$

$$[c_j] = \begin{bmatrix} 200 \\ 1200 \\ 400 \\ 2400 \\ 250 \end{bmatrix},$$

$$[d_j] = \begin{bmatrix} 10 \\ 5 \\ 10 \\ 8 \\ 6 \end{bmatrix}$$

The optimal solution (x^*, y^*) of the LP relaxation can never give a tight dual bound, because the big-M constraint $\sum_{i=1}^m x^*_{ij} \le m y^*_j$ and the optimality of y^* imply that

$$y^*_j = \frac{1}{m} \sum_{i=1}^m x^*_{ij}$$

is always fractional, unless all $x_{ij} = 1$ for the same index j.

Variable fixing: $\gamma_{ij} \ge \gamma_{i0} \Rightarrow x_{ij} = 0$. This removes all variables x_{ij} apart from $x_{10}, \dots, x_{60}$, x_{11}, x_{21}, x_{22}, x_{43}, x_{53}, x_{54}, x_{45}, x_{55}, x_{65}, and it allows to tighten the big-M constraints to

$$\sum_{i=1}^m x^*_{ij} \le |I_j| y_j,$$

where $|I_j| = \{i : \gamma_{ij} < \gamma_{i0}\}$.

Now solving the LP relaxation yields the dual bound 8,940 and the following non-zero values $x^*_{20} = 6/10$, $x^*_{30} = 1$, $y^*_1 = 7/10$, $x^*_{11} = 1$, $x^*_{21} = 4/10$, $y^*_3 = 1$, $x^*_{43} = x^*_{53} = 1$, $y^*_5 = 1/3$, $x^*_{65} = 1$.

In a similar discussion of the UFL we previously noticed that the big-M constraint $\sum_{i=1}^m x_{ij} \le |I_j| y_j$ may be replaced by the strong formulation *given by the valid inequalities*

$$\text{Class C1:} \quad x_{ij} \le y_j, \quad (j \in [1, n];\ i \in I_j).$$

Rather than imposing all of these constraints at each node of the branch-and-cut algorithm, we only use these as cuts when needed. In our case, the valid inequality $x_{11} \le y_1$ is a cut of x^*, because $x^*_{11} = 1 > y^*_1$.

The separation problem associated with class C1 is straightforward to solve via enumeration and scanning. In addition to the cut $x_{11} \leq y_1$, we also find the cut $x_{65} \leq y_5$.

After adding the two cuts to the formulation, we re-optimise the LP solution via dual simplex pivots and obtain the dual bound $9,900$ and the following nonzero components: $x^*_{30} = 1$, $x^*_{60} = 3/4$, $y^*_1 = 1$, $x^*_{11} = x^*_{21} = 1$, $y^*_3 = 3/4$, $x^*_{43} = 3/4$, $x^*_{53} = 3/4$, $y^*_5 = 1/4$, $x^*_{45} = x^*_{55} = x^*_{65} = 1$.

None of the inequalities of class C1 are violated. However, note that $y_1 + y_3 \leq 1$ is a valid inequality, because $d_1 + d_3 > 19$, hence this is a cut for (x^*, y^*). More generally, we consider the following valid inequalities

$$\text{Class C2:} \quad \sum_{j\in S} y_j \leq |S| - 1, \quad \forall\, S \subseteq [1,n] \ \text{s.t.} \sum_{j\in S} d_j > D.$$

To solve the separation problem associated with class C2, we need to find the indicator vector $z \in \{0,1\}^n$ of a set S such that

$$\sum_{j=1}^{n} y^*_j z_j = \sum_{j\in S} y^*_j > |S| - 1 = \sum_{j=1}^{n} z_j - 1,$$
$$\sum_{j=1}^{n} d_j z_j = \sum_{j\in S} d_j \geq D + \varepsilon$$

for some sufficiently small ε.

We can find the deepest cut from this class by solving the knapsack problem

$$\begin{aligned} w^* = \min_z \ & \sum_{j=1}^{n} (1 - y^*_j) z_j \\ \text{s.t.} \ & \sum_{j=1}^{n} d_j z_j \geq D + \varepsilon, \\ & z_j \in \{0,1\}, \quad (j \in [1,n]) \end{aligned}$$

and checking if $w^* < 1$. The knapsack problem is quite easy to solve via branch-and-bound and allows for variable fixing $z_j = 1$ if $y^*_j = 1$, which reduces the number of variables.

Adding the cut $y_1 + y_3 \leq 1$ from class C2 to the formulation and re-optimising the LP relaxation via dual simplex iterations yields the dual bound 10,880 and non-zero variables $x^*_{30} = 1$, $y^*_1 = 1$, $x^*_{11} = x^*_{21} = 1$, $y^*_4 = 3/8$, $x^*_{54} = 3/8$, $y^*_5 = 1$, $x^*_{45} = x^*_{65} = 1$, $x_{55} = 5/8$.

Class C1 does not yield any violated inequalities, but from C2 we find the cut $y_1 + y_4 + y_5 \leq 2$. Adding this inequality to the formulation and re-optimising the LP yields the dual bound 11,100 and the non-zero variables $x^*_{30} = 1$, $y^*_1 = 1$, $x^*_{11} = x^*_{21} = 1$, $y^*_5 = 1$, $x^*_{45} = x^*_{55} = x^*_{65} = 1$, which is integral valued and hence (IP)-optimal.

Note that no branching was necessary in the above example. In general, using cuts from classes C1 and C2 in the branch-and-cut framework reduces the number of nodes generated by 2 or 3 orders of magnitude over straightforward branch-and-bound when applied to larger search index construction problems.

Class C2 can be generalised to other binary programming problems, where they are known as *cover inequalities*.

16.3 Cover Inequalities

Definition 16.5 (Cover Inequality). *Let $a^{\mathrm{T}}x \le r$ with $a \in \mathbb{R}^n_+, r \in \mathbb{R}_+$ be a constraint on binary decision variables $x_i \in \{0,1\}$. A* cover *of this constraint is a subset $C \subseteq \{1,\dots,n\}$ such that $\sum_{j\in C} a_j > r$.*

A cover is minimal *if no strict subset $S \subsetneq C$ is a cover.*

The cover inequality associated with a cover C is the inequality $\sum_{j\in C} x_j \le |C|-1$.

Proposition 16.6. *i) The cover inequality $\sum_{j\in C} x_j \le |C|-1$ associated with a cover C of the constraint $a^{\mathrm{T}}x \le r$ is a valid inequality for $\{x \in \{0,1\}^n : a^{\mathrm{T}}x \le r\}$.*

ii) If $S \subset C$ is also a cover, then the cover inequality $\sum_{j\in S} x_j \le |S|-1$ is stronger than the cover inequality associated with C.

iii) Only cover inequalities associated with minimal covers are essential.

Proof. i) Suppose to the contrary that $\exists\, x \in \{0,1\}^n$ such that $a^{\mathrm{T}}x \le r$ and $\sum_{j\in C} x_j \ge |C|$. Since x is binary, this implies $x_j = 1\ \forall\, j \in C$, and then, using $a_j \ge 0$,

$$a^{\mathrm{T}}x \ge \sum_{j\in C} a_j x_j = \sum_{j\in C} a_j > r. \quad \lightning$$

ii) We need to show that for $x \in \{0,1\}^n$, $\sum_{j\in S} x_j \le |S|-1$ implies $\sum_{j\in C} x_j \le |C|-1$ (but not necessarily the other way round). Let $S_0^c = \{j \in C\setminus S : x_j = 0\}$ and $S_1^c = \{j \in C \setminus S : x_j = 1\}$. Then

$$\sum_{j\in C} x_j = \sum_{j\in S} x_j + \sum_{j\in S_1^c} x_j \le |S| - 1 + |S_1^c| \le |C| - 1.$$

iii) If the cover C is not minimal, then there exists a cover $S \subsetneq C$, and by Part ii) the cover inequality associated with S renders the inequality $\sum_{j\in C} x_j \le |C| - 1$ redundant. □

Algorithm 16.7 (Minimal cover separation)**.**

```
Input: $a \in \mathbb{R}^n_+$, $r \in \mathbb{R}_+$, $x^* \in \{0,1\}^n$ such that $a^{\mathrm{T}} x^* > r$;
Output: minimal cover $C$ whose associated cover inequality is a cut for $x^*$;
Initialise: $C = \emptyset$, $\ell = 1$;
compute list $I = \{j_1, \dots, j_k\} = \{j : x^*_j > 0\}$ ordered such that
$a_{j_1} \geq \cdots \geq a_{j_k}$;
while $\ell \leq k$ and $\sum_{j \in C} a_j \leq r$ do
    $C \leftarrow C \cup \{j_\ell\}$;
    $\ell \leftarrow \ell + 1$;
end
```

Definition 16.8 (Extended cover Inequality)**.** *Let $a^{\mathrm{T}} x^* > r$ with $a \in \mathbb{R}^n_+, r \in \mathbb{R}_+$, and let $\sum_{j \in C} x_j \leq |C| - 1$ be the cover inequality with the minimal cover C constructed by Algorithm (Minimal cover separation), and $a^* := a_{j_1} = \max_{j \in C} a_j$. Let $E = \{j : a_j > a^*\}$. The associated* extended cover inequality *is defined as $\sum_{j \in C \cup E} x_j \leq |C| - 1$.*

Note that, although $C \cup E$ is a cover, the extended cover inequality is not the cover inequality associated with $C \cup E$, because the right-hand side is $|C| - 1$, not $|C \cup E| - 1$.

Proposition 16.9. *i) The extended cover inequality $\sum_{j \in C \cup E} x_j \leq |C| - 1$ is a valid inequality for the set $\{x \in \{0,1\}^n : a^{\mathrm{T}} x \leq r\}$.*

ii) The extended cover inequality is stronger than the cover inequality $\sum_{j \in C} x_j \leq |C| - 1$.

Proof. i) Suppose to the contrary that $\exists\, x \in \{0,1\}^n$ such that $a^{\mathrm{T}} x \leq r$ and $\sum_{j \in C \cup E} x_j \geq |C|$. Then x is a binary vector with at least $|C|$ components equal to 1. Using $a_j, x_j \geq 0 \;\forall\, j$ and $a_j > a^* \;\forall\, j \in E$ by construction, we find

$$a^{\mathrm{T}} x \geq \sum_{j \in C \cup E} a_j x_j \geq \sum_{j \in C} a_j > r. \quad \lightning$$

ii) We need to show that for $x \in \mathbb{R}^n_+$, $\sum_{j \in C \cup E} x_j \leq |C| - 1$ implies $\sum_{j \in C} x_j \leq |C| - 1$ (but not necessarily the other way round). This follows trivially from the disjointness of C and E and $x_j \geq 0$. $\square$

16.4 Lifted Cover Inequalities

This section consists of non-examinable material.

Lemma 16.10 (Lifting valid inequalities)**.** *Let $S := \{j_1, \dots, j_t\} \subset [1, n]$, $\alpha_0, \alpha_{j_s} \geq 0$, $(s = 1, \dots, t)$, $a \in \mathbb{R}^n_+$ and $r \in \mathbb{R}_+$ be given such that $\sum_{s=1}^{t} \alpha_{j_s} x_{j_s} \leq \alpha_0$ is a* *valid inequality for the set* $\{x \in \{0,1\}^n : a^{\mathrm{T}} x \le r\}$. *Let* $j_{t+1} \in [1,n] \setminus S$ *and*

$$\begin{aligned} \zeta_{t+1} := \max & \sum_{s=1}^{t} \alpha_{j_s} x_{j_s} \\ \text{s.t.} & \sum_{s=1}^{t} a_{j_s} x_{j_s} + a_{j_{t+1}} \le r, \\ & x_{j_s} \in \{0,1\}, \quad (s = 1,\dots,t). \end{aligned}$$

Then for every $\alpha_{j_{t+1}} \in [0, \alpha_0 - \zeta_{t+1}]$ *the* lifted inequality $\sum_{s=1}^{t+1} \alpha_{j_s} x_{j_s} \le \alpha_0$ *is valid for the set* $\{x \in \{0,1\}^n : a^{\mathrm{T}} x \le r\}$. *Furthermore, the larger* α_{t+1}*, the stronger the inequality.*

Proof. Let $x \in \{0,1\}^n$ be such that $a^{\mathrm{T}} x \le r$. If $x_{j_{t+1}} = 0$, then the lifted inequality follows from the validity of the unlifted inequality. If $x_{j_{t+1}} = 1$, then x satisfies the constraints of the optimisation problem, and hence, $\sum_{s=1}^{t} \alpha_{j_s} x_{j_s} \le \zeta_{t+1}$, with $\zeta_{t+1} \ge 0$ because $\alpha_{j_s}, x_{j_s} \ge 0$, $(s = 1,\dots,t)$. Therefore,

$$\sum_{s=1}^{t} \alpha_{j_s} x_{j_s} + \alpha_{j_{t+1}} x_{j_{t+1}} \le \zeta_{t+1} + \alpha_0 - \zeta_{t+1} = \alpha_0.$$

Furthermore, if $0 \le \alpha^{[1]} < \alpha^{[2]} \le \alpha_0 - \zeta_{t+1}$, then for $x \in \mathbb{R}^n_+$, $\sum_{s=1}^{t} \alpha_{j_s} x_{j_s} + \alpha^{[2]} x_{j_{t+1}} \le \alpha_0$ implies $\sum_{s=1}^{t} \alpha_{j_s} x_{j_s} + \alpha^{[1]} x_{j_{t+1}} \le \alpha_0$, but not necessarily the other way round. □

Let $x^* \in \mathbb{R}^n_+$ be such that $a^{\mathrm{T}} x^* > r$ with $a \in \mathbb{R}^n_+, r \in \mathbb{R}_+$. We would like to use a tight lifted cover inequality to cut x^* from the set $\{x \in \{0,1\}^n : a^{\mathrm{T}} x \le r\}$. The following algorithm solves this separation problem:

Algorithm 16.11 (Separation by lifted cover inequalities)**.**

```
using Algorithm (Minimal cover separation), find minimal cover inequality
$\sum_{j\in C} x_j \le |C| - 1$;
fix an ordering $j_1,\dots,j_p$ of $[1,n] \setminus C$;
for $t = 1$ to $p$ do
    using branch & bound, solve knapsack problem
        $\zeta_t := \max \sum_{s=1}^{t-1} \alpha_{j_s} x_{j_s} + \sum_{j\in C} x_j$
        s.t. $\sum_{s=1}^{t-1} a_{j_s} x_{j_s} + \sum_{j\in C} a_j x_j + a_{j_t} \le r$,
        $x_j \in \{0,1\}, \quad (j \in C \cup \{j_s : s \in [1, t-1]\})$;
    $\alpha_{j_t} := |C| - 1 - \zeta_t$;
end
```

Proof. (Proof of Correctness) By construction, the cover inequality $\sum_{j\in C} x_j \leq |C| - 1$ is a cut for x^*, and applying Lemma (Lifting valid inequalities) recursively, it follows that the lifted cover inequality $\sum_{s=1}^{p} \alpha_{j_s} x_{j_s} + \sum_{j\in C} x_j \leq |C| - 1$ is a valid inequality stronger than the cover inequality. Hence, it is also a cut for x^*. □

Example 16.12 (Generalised Assignment Problem by Branch & Cut)**.** *A fairly general class of IPs is the Generalised Assignment Problem (GAP) that takes the following form, which uses a combination of knapsack and assignment constraints:*

$$\begin{aligned}
\textit{(GAP)} \quad & \max_x \sum_{i=1}^{m}\sum_{j=1}^{n} p_{ij} x_{ij} \\
\textit{s.t.} \quad & \sum_{j=1}^{n} c_{ij} x_{ij} \leq b_i, \quad (i = 1, \dots, m) \qquad (26) \\
& \sum_{i=1}^{m} x_{ij} = 1, \quad (j = 1, \dots, n) \qquad (27) \\
& x_{ij} \in \{0, 1\}, \quad (i = 1, \dots, m;\ j = 1, \dots, n).
\end{aligned}$$

GUB/SOS branching (see Lecture 9) on the assignment constraints (27) *ensures that the feasible sets of subproblems are balanced (children of the same parent node have approximately equal cardinality).*

Cuts for optimal solutions x^* of LP relaxed subproblems can be constructed in the form of lifted cover inequalities deriving from the knapsack constraints (26)*.*

The branch & cut algorithm solves the problem in two to three orders of magnitude fewer nodes than the branch & bound approach.