



SAPIENZA
UNIVERSITÀ DI ROMA

Integer Linear Programming

Volume 2 — Exact methods

The cutting plane algorithm with Gomory fractional cuts, LP-based branch-and-bound and branch-and-cut. Examples worked out step by step and exercises with complete solutions.

Fabio Furini

fabiofurini.github.io

Department of Computer, Control and Management Engineering — Sapienza University of Rome

Online version and downloadable material:
fabiofurini.github.io/integer-linear-programming/
github.com/fabiofurini/integer-linear-programming

Academic year 2026–2027

Contents

III	Exact methods	2
6	The cutting plane algorithm with Gomory fractional cuts	3
6.1	Gomory fractional cuts	4
6.2	A cutting-plane algorithm based on Gomory fractional cuts	14
6.3	The car example	27
6.4	Typical mistakes	27
6.5	In summary	28
7	LP-based branch-and-bound	29
7.1	The idea in words and in a picture	30
7.2	Definitions and notation	31
7.3	The algorithm	34
7.4	A short recap of the dual simplex algorithm	37
7.5	Branching-variable selection strategies	38
7.6	Branching-node selection strategies	40
7.7	Worked examples on the car example	42
7.8	Typical mistakes	61
7.9	Extensions	62
7.10	In summary	62
8	Branch-and-cut	64
8.1	Branch-and-cut algorithms	65
8.2	Global and local valid inequalities	65
8.3	Cut-and-branch and branch-and-cut	66
8.4	Worked example: cut-and-branch with Gomory cuts	68
8.5	Local cuts: an example	70
8.6	Practical aspects	72
8.7	Typical mistakes	73
8.8	In summary	74



Part III

Exact methods



The cutting plane algorithm with Gomory fractional cuts

Prerequisites

- From the module *Linear Programming*, [Chapter 10 «Dual Simplex algorithm»](#): simplex dictionaries, dual feasible bases and dual pivoting operations. A short recap is given in this chapter.
- From this module, the chapters «Integer linear programming» and «Primal and dual bounds»: ILP problems, the LP relaxation, and the upper bound given by the LP relaxation.
- From this module, the chapter «Valid inequalities»: valid inequalities and cuts that remove fractional points.

Objectives

At the end of this chapter you will be able to:

- generate a Gomory fractional cut from a row of a simplex dictionary, and prove that it is valid and that it cuts off the current basic solution;
- insert a cut into the dictionary and re-optimize with the dual simplex algorithm;
- execute a cutting plane algorithm by hand, with the largest right-hand side selection strategy and its tie-breaking rules;
- compute the upper bound at every iteration, and explain why the algorithm may need many cuts;
- explain why these cuts are valid only for ILP problems with integer data.

6.1 Gomory fractional cuts

Definition 6.1: cutting plane

Given an ILP problem and a fractional optimal solution $\tilde{\mathbf{x}}(\text{LP})$ of its LP relaxation, a **cutting plane** (or **cut**) is a valid inequality for the ILP problem that cuts off $\tilde{\mathbf{x}}(\text{LP})$.

Gomory fractional cuts are valid inequalities that cut off fractional basic feasible solutions of the LP relaxation of an ILP problem without excluding any integer feasible solution.

- **We consider ILP problems in canonical maximization form with integer input data.** For these problems, the slack variables introduced in the standard-form LP relaxation also take integer values in every integer feasible solution of the ILP problem.
- Gomory fractional cuts are valid only if all the variables, slack variables included, are integer. For MIP problems with continuous variables they can cut off feasible solutions, and a different family of cuts is needed.
- We recall that a simplex dictionary of the LP relaxation in standard form reads as follows

$$\begin{cases} x_i = \bar{b}_i - \sum_{j \in \mathcal{N}} \bar{a}_{ij} x_j, & \forall i \in \mathcal{B}, \\ z = \bar{z} + \sum_{j \in \mathcal{N}} \bar{c}_j x_j. \end{cases}$$

where $\mathcal{B}$ is the set of indices of the basic variables and $\mathcal{N}$ is the set of indices of the non-basic variables. If $\bar{b}_i \geq 0$ for every $i \in \mathcal{B}$, then the dictionary is associated with the following feasible basic solution:

$$x_i = \bar{b}_i, \quad \forall i \in \mathcal{B}, \quad x_j = 0, \quad \forall j \in \mathcal{N}.$$

If, in addition, at least one $\bar{b}_i$ is fractional, then this basic feasible solution is fractional.

- Gomory fractional cuts are obtained from the rows of a simplex dictionary associated with a fractional feasible basic solution of the LP relaxation in standard form of an ILP problem.

Definition 6.2: Gomory fractional cut

Given a row $i \in \mathcal{B}$ of a simplex dictionary associated with a feasible basic solution such that $\bar{b}_i$ is fractional, the linear inequality

$$\sum_{j \in \mathcal{N}} (\bar{a}_{ij} - \lfloor \bar{a}_{ij} \rfloor) x_j \geq \bar{b}_i - \lfloor \bar{b}_i \rfloor \quad (6.1)$$

is called the **Gomory fractional cut** generated by row i .

- We will prove that any Gomory fractional cut does not cut off any integer feasible solution of the ILP problem and that it cuts off the fractional feasible basic solution.

Example 6.1: ILP in canonical form, LP relaxation in standard form and dictionaries

Consider the following ILP problem in canonical maximization form with 2 integer variables and 2 linear constraints (called [Example 1](#)):

$$\begin{aligned} \max \quad & 2x_1 + 4x_2 \\ & -x_1 + x_2 \leq 2 \\ & 8x_1 + 2x_2 \leq 19 \\ & x_1, x_2 \in \mathbb{Z}_{\geq 0} \end{aligned}$$

The LP relaxation in standard form with 4 non-negative variables (2 slack variables) and 2 linear constraints reads as follows:

$$\begin{aligned} \max \quad & 2x_1 + 4x_2 \\ & -x_1 + x_2 + x_3 = 2 \\ & 8x_1 + 2x_2 + x_4 = 19 \\ & x_1, x_2, x_3, x_4 \geq 0 \end{aligned}$$

As an example let us consider the following simplex dictionary associated with a fractional basic feasible solution of the LP relaxation in standard form of the ILP problem:

$$\left\{ \begin{array}{l} x_1 = \frac{3}{2} + \frac{1}{5}x_3 - \frac{1}{10}x_4 \\ x_2 = \frac{7}{2} - \frac{4}{5}x_3 - \frac{1}{10}x_4 \\ z = 17 - \frac{14}{5}x_3 - \frac{3}{5}x_4 \end{array} \right. \quad \text{where } \bar{A} = \begin{pmatrix} -\frac{1}{5} & \frac{1}{10} \\ \frac{4}{5} & \frac{1}{10} \end{pmatrix} \quad \bar{b} = \begin{pmatrix} \frac{3}{2} \\ \frac{7}{2} \end{pmatrix}$$

and $\mathcal{B} = \{1, 2\}$ and $\mathcal{N} = \{3, 4\}$. The associated fractional basic feasible solution is:

$$\bar{x}_1 = \frac{3}{2}, \quad \bar{x}_2 = \frac{7}{2}, \quad \bar{x}_3 = 0, \quad \bar{x}_4 = 0 \quad \text{of value} \quad \bar{z} = 17$$

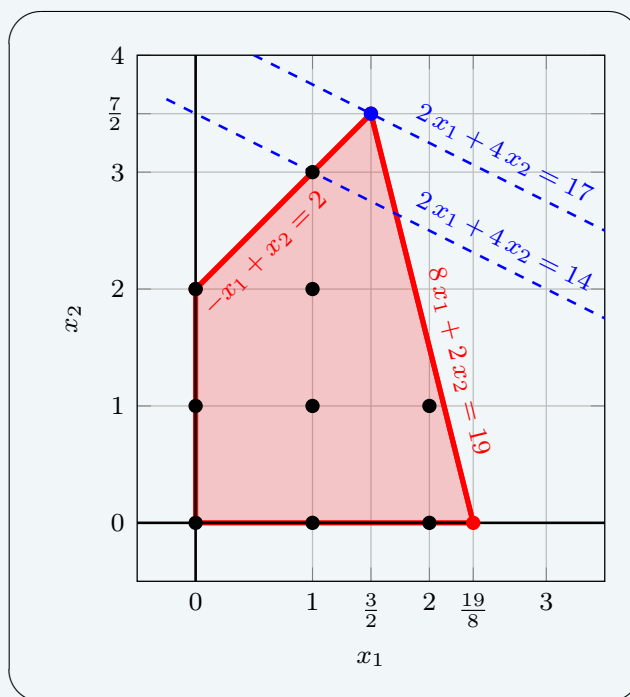
Example 6.2: Gomory fractional cuts

Let us consider again the ILP problem of [Example 1](#) in canonical maximization form with 2 integer variables and 2 linear constraints:

$$\begin{aligned}
\max \quad & 2x_1 + 4x_2 \\
& -x_1 + x_2 \leq 2 \\
& 8x_1 + 2x_2 \leq 19 \\
& x_1, x_2 \in \mathbb{Z}_{\geq 0}
\end{aligned}$$

and its LP relaxation in canonical form with 2 non-negative variables and 2 linear constraints:

$$\begin{aligned}
\max \quad & 2x_1 + 4x_2 \\
& -x_1 + x_2 \leq 2 \\
& 8x_1 + 2x_2 \leq 19 \\
& x_1, x_2 \geq 0
\end{aligned}$$



An optimal solution $\tilde{\mathbf{x}}(\text{ILP})$ of the ILP problem is

$$\tilde{x}_1 = 1, \quad \tilde{x}_2 = 3$$

The optimal value is $z(\text{ILP}) = 14$. An optimal solution $\tilde{\mathbf{x}}(\text{LP})$ of the LP relaxation is

$$\tilde{x}_1 = \frac{3}{2}, \quad \tilde{x}_2 = \frac{7}{2}$$

The optimal value is $z(\text{LP}) = 17$.

Let us consider the following simplex dictionary associated with a fractional basic feasible solution of the LP relaxation in standard form of the ILP problem of [Example 1](#):

$$\begin{cases} x_1 = \frac{3}{2} & + & \frac{1}{5} x_3 & - & \frac{1}{10} x_4 \\ x_2 = \frac{7}{2} & - & \frac{4}{5} x_3 & - & \frac{1}{10} x_4 \\ z = 17 & - & \frac{14}{5} x_3 & - & \frac{3}{5} x_4 \end{cases} \quad \text{where} \quad \bar{A} = \begin{pmatrix} -\frac{1}{5} & \frac{1}{10} \\ \frac{4}{5} & \frac{1}{10} \end{pmatrix} \quad \bar{b} = \begin{pmatrix} \frac{3}{2} \\ \frac{7}{2} \end{pmatrix}$$

- The Gomory fractional cut generated by the first row is:

$$\underbrace{\left(-\frac{1}{5} - \left\lfloor -\frac{1}{5} \right\rfloor\right)}_{-\frac{1}{5} - (-1) = \frac{4}{5}} x_3 + \underbrace{\left(\frac{1}{10} - \left\lfloor \frac{1}{10} \right\rfloor\right)}_{\frac{1}{10} - 0 = \frac{1}{10}} x_4 \geq \underbrace{\left(\frac{3}{2} - \left\lfloor \frac{3}{2} \right\rfloor\right)}_{\frac{3}{2} - 1 = \frac{1}{2}}$$

$$\frac{4}{5} x_3 + \frac{1}{10} x_4 \geq \frac{1}{2} \quad \Rightarrow \quad 8 x_3 + x_4 \geq 5$$

From the standard form of the LP relaxation, we have:

$$x_3 = 2 + x_1 - x_2 \quad \text{and} \quad x_4 = 19 - 8 x_1 - 2 x_2$$

Substituting in the Gomory fractional cut, we can express it in terms of the original variables of the ILP problem:

$$8 (2 + x_1 - x_2) + (19 - 8 x_1 - 2 x_2) \geq 5 \quad \Rightarrow \quad x_2 \leq 3$$

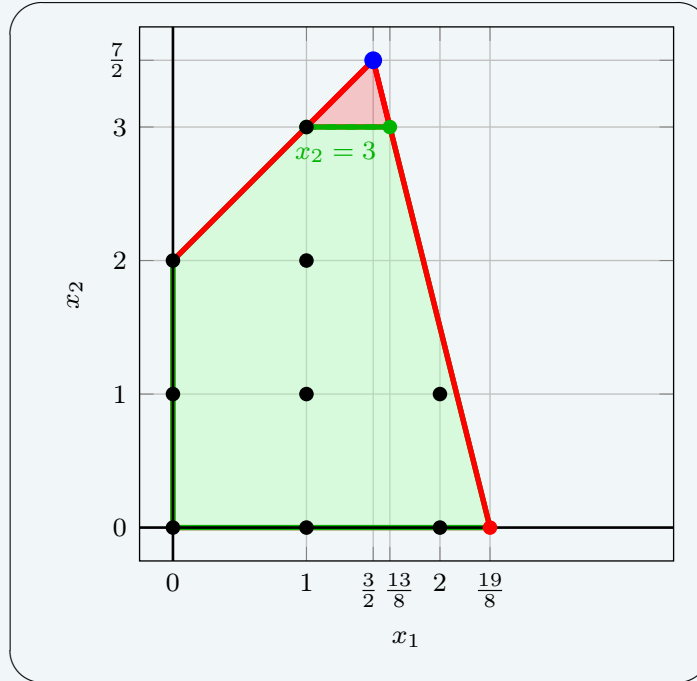
- The Gomory fractional cut generated by the second row is:

$$\underbrace{\left(\frac{4}{5} - \left\lfloor \frac{4}{5} \right\rfloor\right)}_{\frac{4}{5} - 0 = \frac{4}{5}} x_3 + \underbrace{\left(\frac{1}{10} - \left\lfloor \frac{1}{10} \right\rfloor\right)}_{\frac{1}{10} - 0 = \frac{1}{10}} x_4 \geq \underbrace{\left(\frac{7}{2} - \left\lfloor \frac{7}{2} \right\rfloor\right)}_{\frac{7}{2} - 3 = \frac{1}{2}}$$

$$\frac{4}{5} x_3 + \frac{1}{10} x_4 \geq \frac{1}{2} \quad \Rightarrow \quad 8 x_3 + x_4 \geq 5$$

This Gomory fractional cut is the same as the one generated from the first row.

Effect of the Gomory fractional cut on the feasible region of the LP relaxation in canonical form.



A Gomory fractional cut can also be generated by the objective-function row of the simplex dictionary.

- Consider the objective-function row of a simplex dictionary:

$$z = \bar{z} + \sum_{j \in \mathcal{N}} \bar{c}_j x_j$$

This row can be rewritten in the same form as the other dictionary rows:

$$z = \bar{z} - \sum_{j \in \mathcal{N}} (-\bar{c}_j) x_j$$

The Gomory fractional cut generated by the objective-function row is

$$\sum_{j \in \mathcal{N}} (-\bar{c}_j - \lfloor -\bar{c}_j \rfloor) x_j \geq \bar{z} - \lfloor \bar{z} \rfloor \quad (6.2)$$

Example 6.3: Gomory fractional cuts

Let us consider the following simplex dictionary associated with a fractional basic feasible solution of the LP relaxation in standard form of the ILP problem of [Example 1](#):

$$\begin{cases} x_1 = \frac{19}{8} - \frac{1}{4} x_2 - \frac{1}{8} x_4 \\ x_3 = \frac{35}{8} - \frac{5}{4} x_2 - \frac{1}{8} x_4 \\ z = \frac{19}{4} + \frac{7}{2} x_2 - \frac{1}{4} x_4 \end{cases} \quad \text{where} \quad \bar{A} = \begin{pmatrix} \frac{1}{4} & \frac{1}{8} \\ \frac{5}{4} & \frac{1}{8} \end{pmatrix} \quad \bar{b} = \begin{pmatrix} \frac{19}{8} \\ \frac{35}{8} \end{pmatrix}$$

- This dictionary is not optimal ($\bar{c}_2 = \frac{7}{2} > 0$): a Gomory fractional cut can be generated from any dictionary with a fractional basic solution, optimal or not. In the figure the basic solution $(\frac{19}{8}, 0)$ is a vertex of the feasible region, not the optimum of the LP relaxation.
- The Gomory fractional cut generated by the first row is:

$$\underbrace{\left(\frac{1}{4} - \left\lfloor \frac{1}{4} \right\rfloor\right)}_{\frac{1}{4} - 0 = \frac{1}{4}} x_2 + \underbrace{\left(\frac{1}{8} - \left\lfloor \frac{1}{8} \right\rfloor\right)}_{\frac{1}{8} - 0 = \frac{1}{8}} x_4 \geq \underbrace{\left(\frac{19}{8} - \left\lfloor \frac{19}{8} \right\rfloor\right)}_{\frac{19}{8} - 2 = \frac{3}{8}}$$

$$\frac{1}{4} x_2 + \frac{1}{8} x_4 \geq \frac{3}{8} \quad \Rightarrow \quad 2 x_2 + x_4 \geq 3$$

From the standard form of the LP relaxation, we have:

$$x_4 = 19 - 8 x_1 - 2 x_2$$

Substituting in the Gomory fractional cut, we can express it in terms of the original variables of the ILP problem:

$$2 x_2 + (19 - 8 x_1 - 2 x_2) \geq 3 \quad \Rightarrow \quad x_1 \leq 2$$

- The Gomory fractional cut generated by the second row is:

$$\underbrace{\left(\frac{5}{4} - \left\lfloor \frac{5}{4} \right\rfloor\right)}_{\frac{5}{4} - 1 = \frac{1}{4}} x_2 + \underbrace{\left(\frac{1}{8} - \left\lfloor \frac{1}{8} \right\rfloor\right)}_{\frac{1}{8} - 0 = \frac{1}{8}} x_4 \geq \underbrace{\left(\frac{35}{8} - \left\lfloor \frac{35}{8} \right\rfloor\right)}_{\frac{35}{8} - 4 = \frac{3}{8}}$$

$$\frac{1}{4} x_2 + \frac{1}{8} x_4 \geq \frac{3}{8} \quad \Rightarrow \quad 2 x_2 + x_4 \geq 3$$

This Gomory fractional cut is the same as the one generated from the first row.

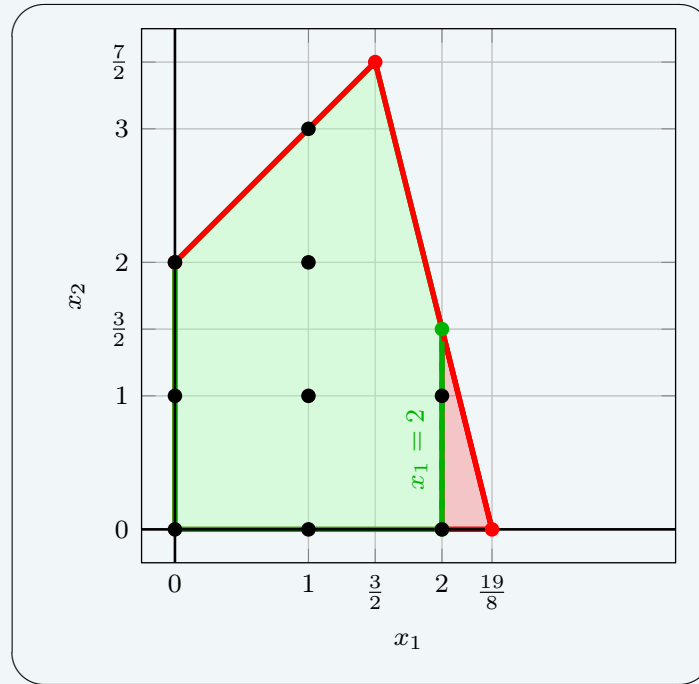
- The Gomory fractional cut generated by the objective-function row is:

$$\underbrace{\left(-\frac{7}{2} - \left\lfloor -\frac{7}{2} \right\rfloor\right)}_{-\frac{7}{2} - (-4) = \frac{1}{2}} x_2 + \underbrace{\left(\frac{1}{4} - \left\lfloor \frac{1}{4} \right\rfloor\right)}_{\frac{1}{4} - 0 = \frac{1}{4}} x_4 \geq \underbrace{\left(\frac{19}{4} - \left\lfloor \frac{19}{4} \right\rfloor\right)}_{\frac{19}{4} - 4 = \frac{3}{4}}$$

$$\frac{1}{2} x_2 + \frac{1}{4} x_4 \geq \frac{3}{4} \quad \Rightarrow \quad 2 x_2 + x_4 \geq 3$$

This Gomory fractional cut is the same as the one generated from the first row.

Effect of the Gomory fractional cut on the feasible region of the LP relaxation in canonical form.



6.1.1 Validity of the Gomory fractional cuts

Observation 6.3

Any Gomory fractional cut cuts off the fractional basic feasible solution associated with the simplex dictionary from which it is generated.

Proof. In the basic feasible solution associated with the dictionary, all non-basic variables are equal to zero:

$$x_j = 0, \quad \forall j \in \mathcal{N}.$$

Now consider the row $i \in \mathcal{B}$ from which the Gomory fractional cut is generated. By definition, $\bar{b}_i$ is fractional. The corresponding Gomory fractional cut is

$$\sum_{j \in \mathcal{N}} (\bar{a}_{ij} - \lfloor \bar{a}_{ij} \rfloor) x_j \geq \bar{b}_i - \lfloor \bar{b}_i \rfloor.$$

At the basic feasible solution, the left-hand side is equal to zero, since $x_j = 0$ for all $j \in \mathcal{N}$. On the other hand,

$$\bar{b}_i - \lfloor \bar{b}_i \rfloor > 0,$$

because $\bar{b}_i$ is fractional. Hence, at the current basic feasible solution, the cut becomes

$$0 \not\geq \bar{b}_i - \lfloor \bar{b}_i \rfloor.$$

Therefore, the Gomory fractional cut cuts off the basic feasible solution associated with the dictionary from which it is generated. $\square$

Observation 6.4

Every integer feasible solution of the ILP problem satisfies every Gomory fractional cut generated from the rows of a simplex dictionary.

Proof. Let us consider a Gomory fractional cut generated from the following row $i \in \mathcal{B}$ of a simplex dictionary:

$$x_i = \bar{b}_i - \sum_{j \in \mathcal{N}} \bar{a}_{ij} x_j. \quad (\text{A})$$

We prove that any integer feasible solution of the ILP problem satisfying this row also satisfies the corresponding Gomory fractional cut.

Since $x_j \geq 0$ for all $j \in \mathcal{N}$ and $\lfloor \bar{a}_{ij} \rfloor \leq \bar{a}_{ij}$, from (A) we obtain

$$x_i \leq \bar{b}_i - \sum_{j \in \mathcal{N}} \lfloor \bar{a}_{ij} \rfloor x_j.$$

Equivalently,

$$x_i + \sum_{j \in \mathcal{N}} \lfloor \bar{a}_{ij} \rfloor x_j \leq \bar{b}_i.$$

The left-hand side is integer, because the solution is integer, the variables appearing in the row are integer-valued, and the coefficients $\lfloor \bar{a}_{ij} \rfloor$ are integer. Hence,

$$x_i + \sum_{j \in \mathcal{N}} \lfloor \bar{a}_{ij} \rfloor x_j \leq \lfloor \bar{b}_i \rfloor.$$

Using (A), we substitute

$$x_i = \bar{b}_i - \sum_{j \in \mathcal{N}} \bar{a}_{ij} x_j$$

into the previous inequality and get

$$\bar{b}_i - \sum_{j \in \mathcal{N}} \bar{a}_{ij} x_j + \sum_{j \in \mathcal{N}} \lfloor \bar{a}_{ij} \rfloor x_j \leq \lfloor \bar{b}_i \rfloor.$$

Rearranging terms gives

$$\sum_{j \in \mathcal{N}} (\bar{a}_{ij} - \lfloor \bar{a}_{ij} \rfloor) x_j \geq \bar{b}_i - \lfloor \bar{b}_i \rfloor.$$

This is exactly the Gomory fractional cut generated from row i . Therefore, the integer feasible solution satisfies this cut.

Since both the integer feasible solution and the Gomory fractional cut were chosen arbitrarily, every integer feasible solution of the ILP problem satisfies every Gomory fractional cut generated from the rows of the dictionary. $\square$

- The same reasoning applies to the Gomory fractional cut generated from the objective-function row. Indeed, the objective-function row can be written as

$$z = \bar{z} + \sum_{j \in \mathcal{N}} \bar{c}_j x_j = \bar{z} - \sum_{j \in \mathcal{N}} (-\bar{c}_j) x_j.$$

If $\bar{z}$ is fractional, the corresponding Gomory fractional cut is

$$\sum_{j \in \mathcal{N}} (-\bar{c}_j - \lfloor -\bar{c}_j \rfloor) x_j \geq \bar{z} - \lfloor \bar{z} \rfloor.$$

- This cut cuts off the current basic feasible solution: at that solution, all non-basic variables are equal to zero, and therefore the left-hand side is zero, while

$$\bar{z} - \lfloor \bar{z} \rfloor > 0.$$

- The cut is valid for every integer feasible solution of the ILP problem. Indeed, if the original objective function has integer coefficients, then z is integer-valued at every integer feasible solution. Moreover,

$$z + \sum_{j \in \mathcal{N}} \lfloor -\bar{c}_j \rfloor x_j \leq \bar{z}.$$

Since the left-hand side is integer, it follows that

$$z + \sum_{j \in \mathcal{N}} \lfloor -\bar{c}_j \rfloor x_j \leq \lfloor \bar{z} \rfloor.$$

Substituting

$$z = \bar{z} + \sum_{j \in \mathcal{N}} \bar{c}_j x_j$$

and rearranging terms gives exactly

$$\sum_{j \in \mathcal{N}} (-\bar{c}_j - \lfloor -\bar{c}_j \rfloor) x_j \geq \bar{z} - \lfloor \bar{z} \rfloor.$$

6.1.2 Inclusion of Gomory fractional cuts into the simplex dictionary

- A Gomory fractional cut is obtained from a row of the current simplex dictionary. In general, both its coefficients and its right-hand side are fractional.
- To insert the cut into the simplex dictionary we proceed in three steps:
 1. substituting the definitions of the slack variables, we rewrite the cut in the original variables as an inequality $\sum_{j=1}^n a_j x_j \leq b$ with integer a_j and b ;
 2. we make it **primitive**: we divide it by the greatest common divisor g of $a_1, \dots, a_n$ and we round down the right-hand side,

$$\sum_{j=1}^n \frac{a_j}{g} x_j \leq \left\lfloor \frac{b}{g} \right\rfloor;$$

3. only then we add the non-negative slack variable $s = \lfloor b/g \rfloor - \sum_{j=1}^n (a_j/g) x_j$, and we substitute the current rows of the dictionary to obtain the row of s .
- The rounding is valid: the left-hand side is integer for every integer $\mathbf{x}$. The slack variable s is integer in every integer feasible solution of the ILP problem.

First the cut is made primitive, then its slack variable is added. A surplus variable added directly to the fractional cut corresponds to the inequality before rounding, which is weaker when g does not divide b .

Example 6.4: Gomory fractional cut

Consider the ILP problem of [Example 1](#).

- The first Gomory fractional cut in the space of the original variables is the following inequality, already primitive ($g = 1$):

$$x_2 \leq 3.$$

Adding a non-negative slack variable gives

$$x_2 + x_5 = 3, \quad x_5 \geq 0.$$

From the current simplex dictionary we have

$$x_2 = \frac{7}{2} - \frac{4}{5} x_3 - \frac{1}{10} x_4.$$

Substituting and solving for the new variable x_5 , the corresponding dictionary row is

$$x_5 = -\frac{1}{2} + \frac{4}{5} x_3 + \frac{1}{10} x_4.$$

- The second Gomory fractional cut in the space of the original variables is the following inequality, already primitive ($g = 1$):

$$x_1 \leq 2.$$

Adding a non-negative slack variable gives

$$x_1 + x_5 = 2, \quad x_5 \geq 0.$$

From the current simplex dictionary we have

$$x_1 = \frac{19}{8} - \frac{1}{4} x_2 - \frac{1}{8} x_4.$$

Substituting and solving for the new variable x_5 , the corresponding dictionary row is

$$x_5 = -\frac{3}{8} + \frac{1}{4} x_2 + \frac{1}{8} x_4.$$

- In [Example 1](#) the coefficients have no common divisor and nothing is rounded. In the car example of the last sections the second cut in the original variables is $6x_1 + 14x_2 \leq 55$: with $g = 2$ the primitive inequality is $3x_1 + 7x_2 \leq \lfloor 55/2 \rfloor = 27$, and its slack variable is $27 - 3x_1 - 7x_2$.

6.2 A cutting-plane algorithm based on Gomory fractional cuts

Definition 6.5: cutting plane algorithm based on Gomory fractional cuts

A **cutting plane algorithm based on Gomory fractional cuts** is an exact method for solving ILP problems based on the repeated solution of LP relaxations. Starting from the LP relaxation of the ILP problem, the algorithm progressively adds Gomory fractional cuts with the goal of removing fractional LP optimal solutions while preserving all integer feasible solutions:

$\rightsquigarrow$ **cut generation operations**

After each cut generation step, the new tighter LP relaxation is solved again:

$\rightsquigarrow$ **LP re-optimization operations**

- The algorithm iterates between cut generation and LP re-optimization until an optimal integer solution is found, if one exists.
- We consider ILP problems in **canonical maximization form with integer input data**.

6.2.1 Cut generation operations

- The **cut generation operations** generate a sequence of increasingly tighter LP relaxations:

$$LP_1, LP_2, LP_3, \dots$$

Each LP problem is obtained from the previous one by adding one or more ***Gomory fractional cuts***.

- When $k = 1$ we have the first LP problem, denoted by LP_1 , which is the LP relaxation of the original ILP problem. For $k \geq 2$, the LP_k problem is obtained from LP_{k-1} by adding the Gomory fractional cuts generated at iteration $k - 1$.

For each iteration k , if the LP_k problem has an optimal solution, its optimal value is denoted by $z(LP_k)$, and an optimal solution is denoted by $\tilde{x}(LP_k)$.

- At each iteration, the current LP relaxation contains all feasible integer solutions of the original ILP problem. Accordingly we have the following upper bound UB on the optimal value of the ILP problem:

$$z(ILP) \leq \underbrace{\lfloor z(LP_k) \rfloor}_{UB}$$

When a Gomory fractional cut is added, the feasible region of the LP relaxation becomes smaller or remains unchanged.

- The algorithm may remove points that are feasible for the LP relaxation, but it does not remove any feasible integer solution of the original ILP problem. In this sense, each new cut tightens the current LP relaxation without excluding any feasible integer solution of the original ILP problem.

6.2.2 High-level pseudocode

Algorithm

Input: An ILP problem in canonical maximization form with integer input data.

Output: An optimal solution to the ILP problem (if one exists) or a certificate of infeasibility/unboundedness.

Step 0: Initialization

Relax the integrality constraints and solve the LP_1 problem. Set the iteration counter $k \leftarrow 1$.

1. if $\tilde{\mathbf{x}}(LP_1)$ is **integer**, then it is an optimal solution and $z(LP_1)$ is the optimal value of the ILP problem. **STOP**.
2. if the LP_1 problem is **infeasible**, then the ILP problem is infeasible. **STOP**.
3. if the LP_1 problem is **unbounded**, then the ILP problem is either infeasible or unbounded. **STOP**.

Step 1: Cut generation

- (a) Select one or more rows $i \in \mathcal{B}$ of the optimal simplex dictionary of the LP_k problem with fractional $\bar{b}_i$ and generate the associated **Gomory fractional cut(s)**:

$$\sum_{j \in \mathcal{N}} (\bar{a}_{ij} - \lfloor \bar{a}_{ij} \rfloor) x_j \geq \bar{b}_i - \lfloor \bar{b}_i \rfloor.$$

- (b) Rewrite each cut in the original variables in primitive form, add its slack variable, and add it to the LP_k problem: this gives the new LP relaxation LP_{k+1} . Update the iteration counter $k \leftarrow k + 1$.

Step 2: Optimality check

- (a) Solve the LP_k problem.
- (b) If the LP_k problem is **infeasible**, then the ILP problem is infeasible. **STOP**.
- (c) Let $\tilde{\mathbf{x}}(LP_k)$ be an optimal solution of the LP_k problem, and let $z(LP_k)$ be its optimal value. If $\tilde{\mathbf{x}}(LP_k)$ is **integer**, then it is an optimal solution of the ILP problem and $z(LP_k)$ is the optimal value of the ILP problem. **STOP**. Otherwise, go to Step 1.

- During the cut generation step, one can generate more than one Gomory fractional cut: by selecting more than one row of the optimal simplex dictionary with fractional $\bar{b}_i$, or the objective-function row.
- The pseudocode does not say how the LP problems are re-optimized after the insertion of the cuts: the next section explains the use of the dual simplex algorithm.

6.2.3 Re-optimization with the dual simplex algorithm

- A new cut differs from the previous LP_k problem by one constraint, so we do not solve the new problem from scratch: we start from the **optimal dictionary** of the LP_k problem and we re-optimize it with the **dual simplex algorithm**.
- The cut gives a new row with the slack variable x_s of the primitive inequality in the basis. Its value in the current basic solution is negative, since the cut removes this solution. The reduced costs do not change.

After the insertion of a cut the dictionary is **dual feasible** but not primal feasible: this is exactly the starting point of the dual simplex algorithm.

Dual pivoting operation on a dictionary $x_i = \bar{b}_i - \sum_{j \in \mathcal{N}} \bar{a}_{ij} x_j$, for $i \in \mathcal{B}$, and $z = \bar{z} + \sum_{j \in \mathcal{N}} \bar{c}_j x_j$, with $\bar{c}_j \leq 0$ for all $j \in \mathcal{N}$:

- a. the **pivot row** is the basic variable with negative value having the smallest index:

$$r = \min \{ i \in \mathcal{B} : \bar{b}_i < 0 \};$$

- b. if $\bar{a}_{rj} \geq 0$ for all $j \in \mathcal{N}$, the LP problem is **infeasible**;

- c. otherwise the **pivot column** x_p attains the minimum ratio (ties: smallest index)

$$\frac{\bar{c}_p}{\bar{a}_{rp}} = s_m = \min \left\{ \frac{\bar{c}_j}{\bar{a}_{rj}} : j \in \mathcal{N}, \bar{a}_{rj} < 0 \right\};$$

- d. after the pivoting operation the objective value is $\bar{z} + s_m \bar{b}_r \leq \bar{z}$.

The algorithm stops when all the basic variables are non-negative: the dictionary is then optimal.

- If the new LP_{k+1} problem is infeasible, the dual simplex algorithm detects it in step b, and the ILP problem is infeasible. Usually one or two pivoting operations are enough.

Example 6.5: cutting plane algorithm

We now describe the execution of the cutting-plane algorithm for the ILP problem of [Example 1](#) using:

1. one Gomory fractional cut at each iteration;
2. the largest right-hand side as the cut selection strategy, with the tie-breaking rules of the section on cut selection (smallest index of the basic variable, objective-function row last);
3. every LP_k problem is re-optimized with the dual simplex algorithm.

Initialization

For the LP_1 problem in canonical form, we have:

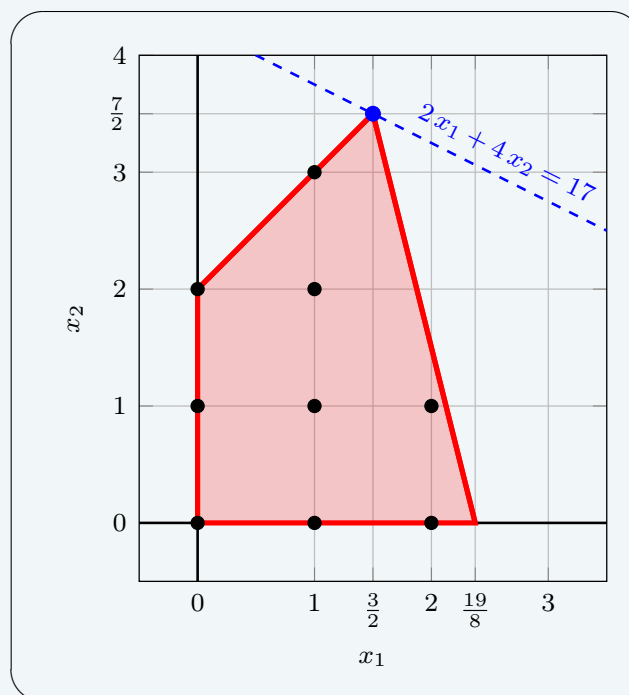
$$\begin{array}{rclcl}
 \max & 2 & x_1 & + & 4 & x_2 \\
 & - & x_1 & + & & x_2 & \leq & 2 \\
 & 8 & x_1 & + & 2 & x_2 & \leq & 19 \\
 & & x_1, & & x_2 & \geq & 0
 \end{array}$$

An optimal solution $\tilde{\mathbf{x}}(\text{LP}_1)$ is:

$$\tilde{x}_1 = \frac{3}{2}, \quad \tilde{x}_2 = \frac{7}{2}$$

The optimal value is:

$$z(\text{LP}_1) = 17$$



The optimal solution $\tilde{\mathbf{x}}(\text{LP}_1)$ is **fractional** and the LP_1 problem is **feasible**.

We have the upper bound on the optimal value of the ILP problem:

$$z(\text{ILP}) \leq 17$$

Iteration 1

a) Cut generation phase

We consider the following simplex dictionary associated with a fractional basic feasible solution of the LP_1 problem in standard form:

$$\begin{cases} x_1 = \frac{3}{2} + \frac{1}{5} x_3 - \frac{1}{10} x_4 \\ x_2 = \frac{7}{2} - \frac{4}{5} x_3 - \frac{1}{10} x_4 \\ z = 17 - \frac{14}{5} x_3 - \frac{3}{5} x_4 \end{cases} \quad \text{where} \quad \bar{A} = \begin{pmatrix} -\frac{1}{5} & \frac{1}{10} \\ \frac{4}{5} & \frac{1}{10} \end{pmatrix} \quad \bar{b} = \begin{pmatrix} \frac{3}{2} \\ \frac{7}{2} \end{pmatrix}$$

- The Gomory fractional cut generated by the first row is:

$$\underbrace{\left(-\frac{1}{5} - \left\lfloor -\frac{1}{5} \right\rfloor\right)}_{-\frac{1}{5} - (-1) = \frac{4}{5}} x_3 + \underbrace{\left(\frac{1}{10} - \left\lfloor \frac{1}{10} \right\rfloor\right)}_{\frac{1}{10} - 0 = \frac{1}{10}} x_4 \geq \underbrace{\left(\frac{3}{2} - \left\lfloor \frac{3}{2} \right\rfloor\right)}_{\frac{3}{2} - 1 = \frac{1}{2}}$$

$$\frac{4}{5} x_3 + \frac{1}{10} x_4 \geq \frac{1}{2} \Rightarrow 8 x_3 + x_4 \geq 5$$

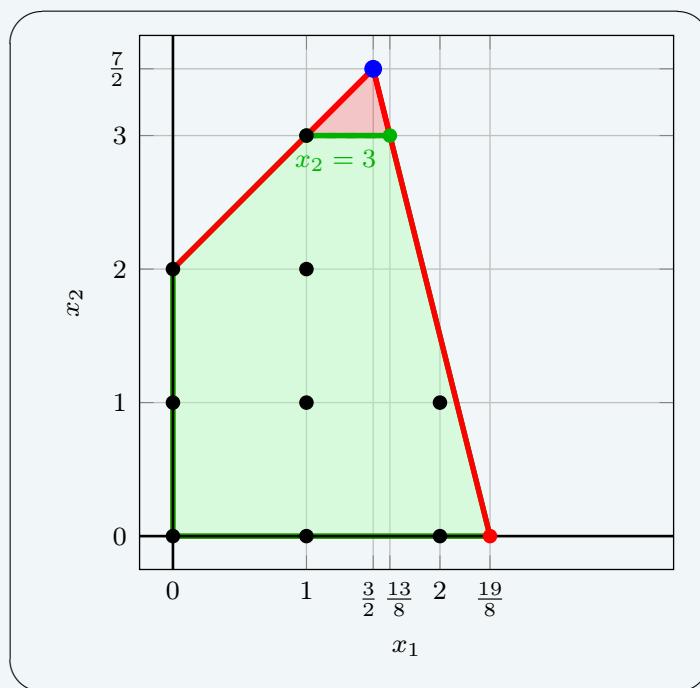
From the standard form of the LP_1 problem, we have:

$$x_3 = 2 + x_1 - x_2 \quad \text{and} \quad x_4 = 19 - 8 x_1 - 2 x_2$$

Substituting in the Gomory fractional cut, we can express it in terms of the original variables of the ILP problem:

$$8(2 + x_1 - x_2) + (19 - 8 x_1 - 2 x_2) \geq 5 \Rightarrow x_2 \leq 3$$

Effect of the Gomory fractional cut generated by the first row on the feasible region of the LP_1 problem in canonical form:



- The Gomory fractional cut generated by the second row is:

$$\underbrace{\left(\frac{4}{5} - \left\lfloor \frac{4}{5} \right\rfloor\right)}_{\frac{4}{5} - 0 = \frac{4}{5}} x_3 + \underbrace{\left(\frac{1}{10} - \left\lfloor \frac{1}{10} \right\rfloor\right)}_{\frac{1}{10} - 0 = \frac{1}{10}} x_4 \geq \underbrace{\left(\frac{7}{2} - \left\lfloor \frac{7}{2} \right\rfloor\right)}_{\frac{7}{2} - 3 = \frac{1}{2}}$$

$$\frac{4}{5} x_3 + \frac{1}{10} x_4 \geq \frac{1}{2} \quad \Rightarrow \quad 8 x_3 + x_4 \geq 5$$

This Gomory fractional cut is the same as the one generated from the first row.

The two cuts are equal and have the largest right-hand side, $\frac{1}{2}$. We select the one generated by the row of x_1 (smallest index) and we create the LP_2 problem by adding it to the LP_1 problem.

b) LP re-optimization phase

For the LP_2 problem in canonical form, we have:

$$\begin{array}{rclcl} \max & 2 & x_1 & + & 4 & x_2 & & & \\ & & - & x_1 & + & & x_2 & \leq & 2 \\ & & 8 & x_1 & + & 2 & x_2 & \leq & 19 \\ & & & & & & x_2 & \leq & 3 \\ & & & & & & x_1, & x_2 & \geq & 0 \end{array}$$

- In the original variables the cut is $x_2 \leq 3$, already primitive (its coefficients have no common divisor).
- The slack variable $x_5 = 3 - x_2 \geq 0$ of the primitive inequality is added. Substituting the current rows of the dictionary, the new row is

$$x_5 = -\frac{1}{2} + \frac{4}{5} x_3 + \frac{1}{10} x_4.$$

- The dictionary below is **dual feasible** (the reduced costs do not change) but not primal feasible, since $\bar{b}_5 = -\frac{1}{2} < 0$:

$$\left\{ \begin{array}{rclcl} x_1 & = & \frac{3}{2} & + & \frac{1}{5} x_3 & - & \frac{1}{10} x_4 \\ x_2 & = & \frac{7}{2} & - & \frac{4}{5} x_3 & - & \frac{1}{10} x_4 \\ x_5 & = & -\frac{1}{2} & + & \frac{4}{5} x_3 & + & \frac{1}{10} x_4 \\ z & = & 17 & - & \frac{14}{5} x_3 & - & \frac{3}{5} x_4 \end{array} \right.$$

- **Dual pivoting operation.** The pivot row is x_5 , the only basic variable with a negative value. The ratios $\bar{c}_j/\bar{a}_{5j}$ of the non-basic variables with $\bar{a}_{5j} < 0$ are

$$\frac{\bar{c}_3}{\bar{a}_{5,3}} = \frac{-\frac{14}{5}}{-\frac{4}{5}} = \frac{7}{2}, \quad \frac{\bar{c}_4}{\bar{a}_{5,4}} = \frac{-\frac{3}{5}}{-\frac{1}{10}} = 6.$$

- The pivot column is x_3 , with $s_m = \frac{7}{2}$, and the objective value becomes $\bar{z} + s_m \bar{b}_5 = 17 + \frac{7}{2} \cdot \left(-\frac{1}{2}\right) = \frac{61}{4}$. The new dictionary is primal and dual feasible, hence optimal:

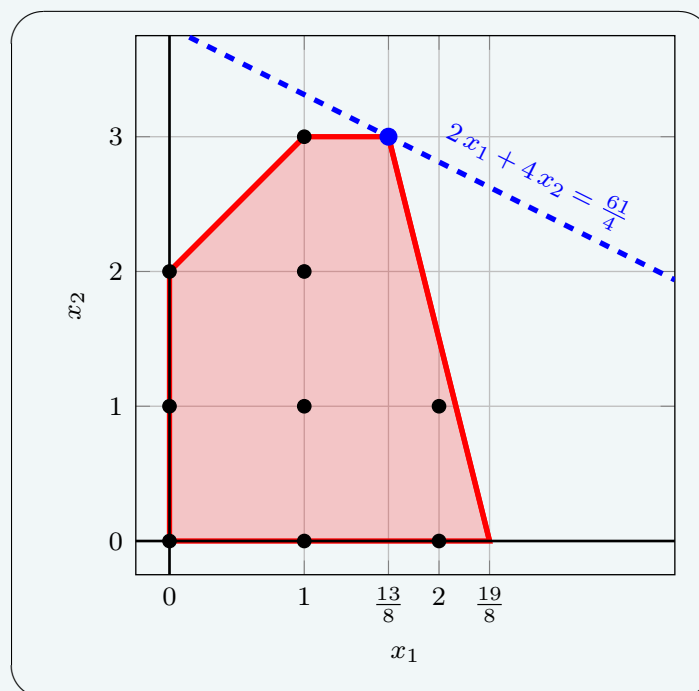
$$\begin{cases} x_1 &= \frac{13}{8} - \frac{1}{8}x_4 + \frac{1}{4}x_5 \\ x_2 &= 3 - x_5 \\ x_3 &= \frac{5}{8} - \frac{1}{8}x_4 + \frac{5}{4}x_5 \\ z &= \frac{61}{4} - \frac{1}{4}x_4 - \frac{7}{2}x_5 \end{cases}$$

An optimal solution $\tilde{\mathbf{x}}(\text{LP}_2)$ is:

$$\tilde{x}_1 = \frac{13}{8}, \quad \tilde{x}_2 = 3$$

The optimal value is:

$$z(\text{LP}_2) = \frac{61}{4}$$



We have the upper bound on the optimal value of the ILP problem:

$$z(\text{ILP}) \leq \underbrace{\left\lfloor \frac{61}{4} \right\rfloor}_{=15}$$

Iteration 2

a) Cut generation phase

We have the following simplex dictionary associated with a fractional basic feasible solution of the LP_2 problem in standard form:

$$\begin{cases} x_1 = \frac{13}{8} - \frac{1}{8} x_4 + \frac{1}{4} x_5 \\ x_2 = 3 - x_5 \\ x_3 = \frac{5}{8} - \frac{1}{8} x_4 + \frac{5}{4} x_5 \\ z = \frac{61}{4} - \frac{1}{4} x_4 - \frac{7}{2} x_5 \end{cases} \quad \text{where } \bar{A} = \begin{pmatrix} \frac{1}{8} & -\frac{1}{4} \\ 0 & 1 \\ \frac{1}{8} & -\frac{5}{4} \end{pmatrix} \quad \bar{b} = \begin{pmatrix} \frac{13}{8} \\ 3 \\ \frac{5}{8} \end{pmatrix}$$

- The Gomory fractional cut generated by the first row is:

$$\underbrace{\left(\frac{1}{8} - \left\lfloor \frac{1}{8} \right\rfloor \right)}_{\frac{1}{8} - 0 = \frac{1}{8}} x_4 + \underbrace{\left(-\frac{1}{4} - \left\lfloor -\frac{1}{4} \right\rfloor \right)}_{-\frac{1}{4} - (-1) = \frac{3}{4}} x_5 \geq \underbrace{\left(\frac{13}{8} - \left\lfloor \frac{13}{8} \right\rfloor \right)}_{\frac{13}{8} - 1 = \frac{5}{8}}$$

$$\frac{1}{8} x_4 + \frac{3}{4} x_5 \geq \frac{5}{8} \quad \Rightarrow \quad x_4 + 6 x_5 \geq 5$$

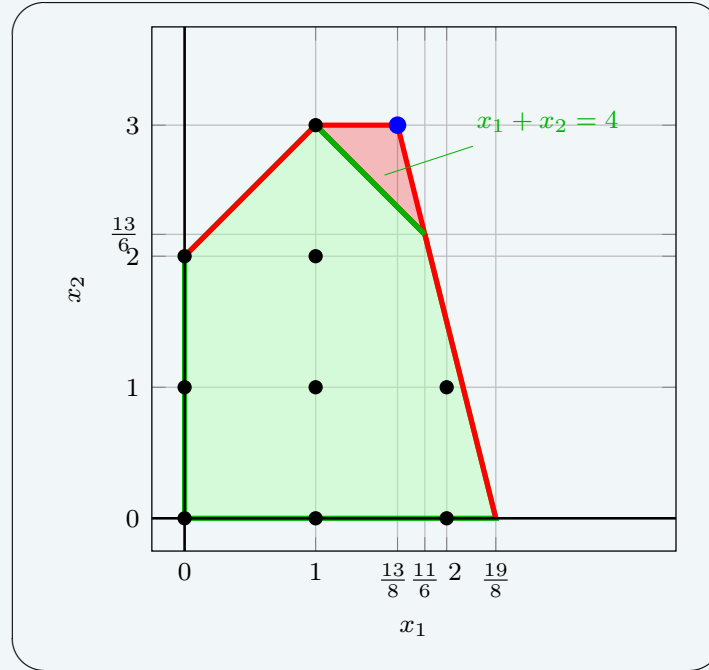
From the standard form of the LP_2 problem, we have:

$$x_4 = 19 - 8x_1 - 2x_2 \quad \text{and} \quad x_5 = 3 - x_2.$$

Substituting in the Gomory fractional cut, we can express it in terms of the original variables of the ILP problem:

$$(19 - 8x_1 - 2x_2) + 6(3 - x_2) \geq 5 \quad \Rightarrow \quad x_1 + x_2 \leq 4.$$

Effect of the Gomory fractional cut generated by the first row on the feasible region of the LP_2 problem in canonical form:



- The Gomory fractional cut generated by the third row is:

$$\underbrace{\left(\frac{1}{8} - \left\lfloor \frac{1}{8} \right\rfloor\right)}_{\frac{1}{8} - 0 = \frac{1}{8}} x_4 + \underbrace{\left(-\frac{5}{4} - \left\lfloor -\frac{5}{4} \right\rfloor\right)}_{-\frac{5}{4} - (-2) = \frac{3}{4}} x_5 \geq \underbrace{\left(\frac{5}{8} - \left\lfloor \frac{5}{8} \right\rfloor\right)}_{\frac{5}{8} - 0 = \frac{5}{8}}$$

$$\frac{1}{8} x_4 + \frac{3}{4} x_5 \geq \frac{5}{8} \quad \Rightarrow \quad x_4 + 6 x_5 \geq 5$$

This Gomory fractional cut is the same as the one generated from the first row.

- The Gomory fractional cut generated by the objective-function row is:

$$\underbrace{\left(\frac{1}{4} - \left\lfloor \frac{1}{4} \right\rfloor\right)}_{\frac{1}{4} - 0 = \frac{1}{4}} x_4 + \underbrace{\left(\frac{7}{2} - \left\lfloor \frac{7}{2} \right\rfloor\right)}_{\frac{7}{2} - 3 = \frac{1}{2}} x_5 \geq \underbrace{\left(\frac{61}{4} - \left\lfloor \frac{61}{4} \right\rfloor\right)}_{\frac{61}{4} - 15 = \frac{1}{4}}$$

$$\frac{1}{4} x_4 + \frac{1}{2} x_5 \geq \frac{1}{4} \quad \Rightarrow \quad x_4 + 2 x_5 \geq 1$$

From the standard form of the LP_2 problem, we have:

$$x_4 = 19 - 8x_1 - 2x_2 \quad \text{and} \quad x_5 = 3 - x_2.$$

Substituting in the Gomory fractional cut, we can express it in terms of the original variables of the ILP problem:

$$(19 - 8x_1 - 2x_2) + 2(3 - x_2) \geq 1 \quad \Rightarrow \quad 2x_1 + x_2 \leq 6.$$

- The slack variable $x_6 = 4 - (x_1 + x_2) \geq 0$ of the primitive inequality is added. Substituting the current rows of the dictionary, the new row is

$$x_6 = -\frac{5}{8} + \frac{1}{8}x_4 + \frac{3}{4}x_5.$$

- The dictionary below is **dual feasible** (the reduced costs do not change) but not primal feasible, since $\bar{b}_6 = -\frac{5}{8} < 0$:

$$\left\{ \begin{array}{rclcl} x_1 & = & \frac{13}{8} & - & \frac{1}{8}x_4 & + & \frac{1}{4}x_5 \\ x_2 & = & 3 & & & - & x_5 \\ x_3 & = & \frac{5}{8} & - & \frac{1}{8}x_4 & + & \frac{5}{4}x_5 \\ x_6 & = & -\frac{5}{8} & + & \frac{1}{8}x_4 & + & \frac{3}{4}x_5 \\ z & = & \frac{61}{4} & - & \frac{1}{4}x_4 & - & \frac{7}{2}x_5 \end{array} \right.$$

- Dual pivoting operation.** The pivot row is x_6 , the only basic variable with a negative value. The ratios $\bar{c}_j/\bar{a}_{6j}$ of the non-basic variables with $\bar{a}_{6j} < 0$ are

$$\frac{\bar{c}_4}{\bar{a}_{6,4}} = \frac{-\frac{1}{4}}{-\frac{1}{8}} = 2, \quad \frac{\bar{c}_5}{\bar{a}_{6,5}} = \frac{-\frac{7}{2}}{-\frac{3}{4}} = \frac{14}{3}.$$

- The pivot column is x_4 , with $s_m = 2$, and the objective value becomes $\bar{z} + s_m \bar{b}_6 = \frac{61}{4} + 2 \cdot \left(-\frac{5}{8}\right) = 14$. The new dictionary is primal and dual feasible, hence optimal:

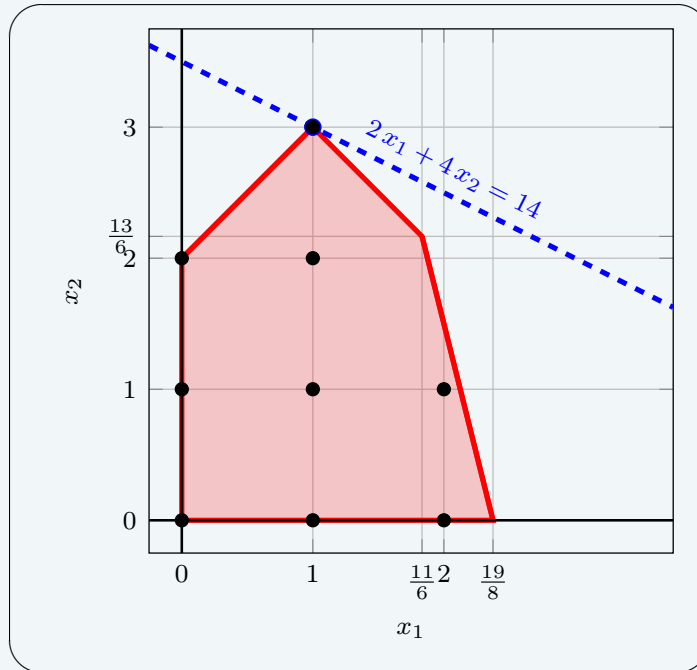
$$\left\{ \begin{array}{rclcl} x_1 & = & 1 & + & x_5 & - & x_6 \\ x_2 & = & 3 & - & x_5 & & \\ x_3 & = & 0 & + & 2x_5 & - & x_6 \\ x_4 & = & 5 & - & 6x_5 & + & 8x_6 \\ z & = & 14 & - & 2x_5 & - & 2x_6 \end{array} \right.$$

An optimal solution $\tilde{\mathbf{x}}(\text{LP}_3)$ is:

$$\tilde{x}_1 = 1, \quad \tilde{x}_2 = 3$$

The optimal value is:

$$z(\text{LP}_3) = 14$$



The optimal solution $\tilde{\mathbf{x}}(\text{LP}_3)$ is **integer**. Accordingly, an optimal solution $\tilde{\mathbf{x}}(\text{ILP})$ is:

$$\tilde{x}_1 = 1, \quad \tilde{x}_2 = 3$$

The optimal value $z(\text{ILP})$ is 14.

6.2.4 Cut selection strategies

- During the cut generation phase, several Gomory fractional cuts can be generated and added to the LP relaxation. The **cut selection strategy** decides which ones.

The **largest right-hand side cut selection strategy** selects the Gomory fractional cut whose fractional right-hand side is maximal among all the candidate cuts.

- The candidate cuts are generated by the rows $i \in \mathcal{B}$ of the dictionary with fractional $\bar{b}_i$; the right-hand side of the cut of row i is

$$\bar{b}_i - \lfloor \bar{b}_i \rfloor.$$

- If the objective value $\bar{z}$ is fractional, the objective-function row is also a candidate, with right-hand side $\bar{z} - \lfloor \bar{z} \rfloor$.
- **Ties.** Cuts with the same right-hand side can be different, so the choice must be fixed to make the execution reproducible: we select the row of the basic variable with the smallest index, and the objective-function row is considered last.

6.2.5 Heuristic initialization

Before executing a cutting-plane algorithm, a **heuristic algorithm** can be run with the goal of finding a high-quality feasible solution to the ILP problem. This heuristic solution provides an initial lower bound LB on the optimal value of the ILP problem.

- At each iteration, the optimal value of the current LP relaxation provides an **upper bound**

$$UB = \lfloor z(LP_k) \rfloor$$

on the optimal value of the ILP problem. As valid cuts are added, this upper bound is progressively improved.

- The availability of a feasible integer solution allows one to monitor the gap between the upper and lower bounds:

$$\frac{UB - LB}{|LB|}$$

The cutting-plane algorithm can be stopped when this gap is zero, then the heuristic solution is proven to be optimal.

6.2.6 Additional remarks

- The finite termination of a cutting plane algorithm with Gomory fractional cuts is guaranteed only with precise rules (a lexicographic dual simplex algorithm and a suitable choice of the rows). With arbitrary rules the algorithm may require a very large number of iterations.
- Gomory fractional cuts are usually very effective in the first iterations, but their effectiveness decreases as the algorithm progresses (**tailing off**); the next section shows an example.
- For this reason, Gomory fractional cuts are often used to strengthen the LP relaxations solved at the nodes of a branch-and-bound tree: the resulting method is a **branch-and-cut algorithm**.

6.3 The car example

- The car example is solved by the cutting plane algorithm with the same rules as in Example 1: one cut per iteration, largest right-hand side, ties by the smallest index of the basic variable and objective-function row last.
- The table reports, for every cut, the inequality in the original variables before rounding (when $g > 1$) and in primitive form, the solution and the value of the next LP problem, and the upper bound $UB = \lfloor z(LP_{k+1}) \rfloor$.

Cut	Before rounding	Primitive cut	$\tilde{\mathbf{x}}(LP_{k+1})$	$z(LP_{k+1})$	UB
1	—	$5x_1 + 14x_2 \leq 52$	$(\frac{211}{60}, \frac{59}{24})$	$\frac{232}{15}$	15
2	$6x_1 + 14x_2 \leq 55$	$3x_1 + 7x_2 \leq 27$	$(\frac{207}{58}, \frac{135}{58})$	$\frac{891}{58}$	15
3	—	$4x_1 + 7x_2 \leq 30$	$(\frac{65}{18}, \frac{20}{9})$	$\frac{275}{18}$	15
4	$7x_1 + 7x_2 \leq 40$	$x_1 + x_2 \leq 5$	$(\frac{25}{6}, \frac{5}{6})$	$\frac{85}{6}$	14
5	—	$9x_1 + 4x_2 \leq 40$	$(4, 1)$	14	14

- The algorithm needs 5 cuts. The upper bound stays at 15 for the first 3 cuts, and the integer solution is found only at the last cut.
- In cuts 2 and 4 the coefficients have a common divisor: the primitive cut is stronger than the inequality before rounding.
- The optimal solution is $\tilde{\mathbf{x}} = (4, 1)$, with value $z(ILP) = 14$.

6.4 Typical mistakes

Warning

- **Using the cut for problems with continuous variables.** The fractional cut is valid only if all the variables and the data are integer: with a continuous variable it may remove feasible solutions.
- **Taking the floor of a coefficient or of a right-hand side.** The cut uses the fractional parts: $\bar{a}_{ij} - \lfloor \bar{a}_{ij} \rfloor$ is always in $[0, 1)$, also for a negative $\bar{a}_{ij}$.
- **Forgetting the objective-function row.** It is a candidate cut if $\bar{z}$ is fractional; with the largest right-hand side strategy it can be the selected one.
- **Adding the surplus variable before making the cut primitive.** The cut in the original variables must first be divided by the greatest common divisor of its coefficients, with the right-hand side rounded down; only then is its slack variable added. Otherwise the cut is weaker.
- **Choosing among equal right-hand sides without a rule.** Different choices give different executions; use the smallest index of the basic variable, and the objective-

function row last.

- **Solving every LP_k problem from scratch.** The answer is the same, but much slower: the dictionary with the new row is dual feasible, and the dual simplex algorithm starts from it.
- **Stopping when the upper bound stops improving.** The algorithm stops only when the solution of the LP_k problem is integer.

6.5 In summary

In summary

- A Gomory fractional cut is generated from a row of the dictionary with fractional $\bar{b}_i$; it cuts off the current basic solution and no integer feasible solution of the ILP problem.
- The cut is valid only for ILP problems with integer data, where slack and surplus variables are integer.
- After each cut the LP problem is re-optimized with the dual simplex algorithm, starting from the dictionary of the previous problem.
- The upper bound is $\lfloor z(LP_k) \rfloor$; the algorithm ends when the LP solution is integer.
- Each cut is rewritten in the original variables, made primitive (divided by the greatest common divisor of its coefficients, right-hand side rounded down), and only then its slack variable is added.
- Cuts work well at the beginning and then lose effectiveness: on the car example 5 cuts are needed.

Dashboard of the car example. The lower and upper bounds on $z(ILP)$ found so far in this module:

Chapter			Lower bound	Upper bound	From
Primal	and	dual	14	15	heuristic solution $(4, 1)$, and $\lfloor z(LP) \rfloor$ with $z(LP) = \frac{31}{2}$
This chapter			14	14	5 Gomory fractional cuts

LP-based branch-and-bound

Prerequisites

- From the module *Linear Programming*, [Chapter 10 «Dual Simplex algorithm»](#): simplex dictionaries, dual feasible bases and dual pivoting operations. A short recap is given in this chapter.
- From this module, the chapter «Integer linear programming»: ILP problems, feasible and optimal solutions, and the car example.
- From this module, the chapter «Primal and dual bounds»: the LP relaxation, lower bounds from feasible solutions, upper bounds from the LP relaxation, and the optimality gap.

Objectives

At the end of this chapter you will be able to:

- explain how branching operations divide a ILP problem into smaller LP problems, and why no feasible solution is lost;
 - apply the three pruning criteria, and decide when the bound of a node can be rounded down;
 - execute an LP-based branch-and-bound algorithm by hand, re-optimizing every node with the dual simplex algorithm;
 - compute the lower bound, the upper bound and the optimality gap at every iteration;
 - compare different branching-variable and branching-node selection strategies on the same instance.
- This chapter defines the LP-based branch-and-bound algorithm for ILP problems in canonical maximization form with integer data, with its branching and bounding operations.
 - The guiding instance is the car example, $\max\{3x_1 + 2x_2 : 10x_1 + 4x_2 \leq 45, 4x_1 + 14x_2 \leq 49, x_1, x_2 \in \mathbb{Z}_{\geq 0}\}$, whose LP relaxation has the fractional optimal solution $(\frac{7}{2}, \frac{5}{2})$, with $z(\text{LP}) = \frac{31}{2}$.

7.1 The idea in words and in a picture

- The **LP-based branch-and-bound algorithm (B&B algorithm)**, for short) is an exact method for ILP problems. It is called **LP-based** because it solves LP relaxations to guide the search for an optimal solution.
- In the optimal solution of the LP relaxation of the car example the variable x_1 takes a fractional value between 3 and 4. Every feasible solution of the ILP problem satisfies either $x_1 \leq 3$ or $x_1 \geq 4$, so we can split the problem into two smaller LP problems, one for each side, and remove the strip in between, which contains no integer point:

↪ **branching operations.**

- Each new LP problem is solved. Its optimal value bounds from above the value of every solution in its region, so it tells us whether the region must be explored further or can be discarded:

↪ **bounding operations.**

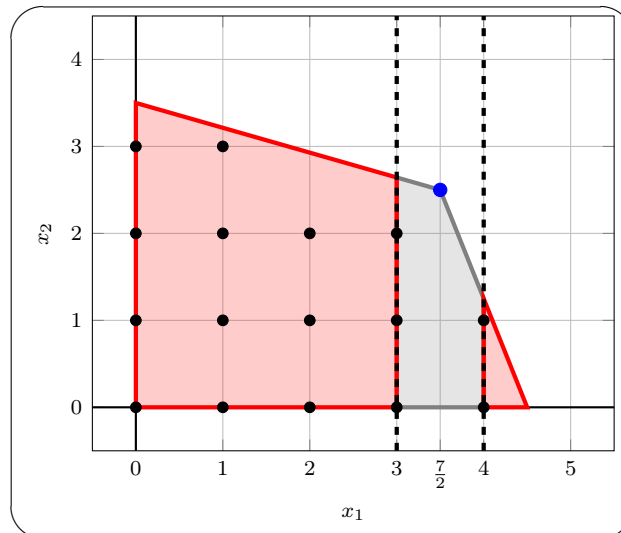


Figure 7.1: Branching on x_1 at the root of the car example: the two red regions are the feasible regions of the children ($x_1 \leq 3$ and $x_1 \geq 4$); the gray strip $3 < x_1 < 4$ is cut off, and it contains no integer point.

A branch-and-bound algorithm explores implicitly all the feasible solutions: it divides the feasible region into smaller regions (branching) and discards every region that cannot contain a better solution than the best one found so far (bounding).

7.2 Definitions and notation

7.2.1 ILP problems in canonical maximization form

- Given n values $c_j \in \mathbb{Z}$, $m \cdot n$ values $a_{ij} \in \mathbb{Z}$ and m values $b_i \in \mathbb{Z}$, an **integer linear programming** (ILP) problem in canonical maximization form, with n integer variables and integer data, reads as follows:

$$\max \sum_{j=1}^n c_j x_j \quad (7.1a)$$

$$\text{subject to } \sum_{j=1}^n a_{ij} x_j \leq b_i, \quad \forall i \in \{1, 2, \dots, m\}, \quad (7.1b)$$

$$x_j \in \mathbb{Z}_{\geq 0}, \quad \forall j \in \{1, 2, \dots, n\}. \quad (7.1c)$$

- The car example is an ILP problem with $n = 2$ and $m = 2$.
- We denote by $z(\text{ILP})$ the optimal value of the problem (7.1), and by $\tilde{\mathbf{x}}(\text{ILP})$ an optimal solution.
- The **LP relaxation** of the problem (7.1), called the LP_1 problem, is obtained by replacing the integrality constraints (7.1c) with the non-negativity constraints $x_j \geq 0$, for all $j \in \{1, 2, \dots, n\}$.

7.2.2 Branch-and-bound tree, incumbent solution and lower bound

The **branching operations** generate a **branch-and-bound tree** containing LP problems. Each LP problem is obtained from the LP relaxation by adding specific **branching constraints**.

- The recursive structure of a B&B algorithm induces the tree. Each **branching node** of the tree corresponds to an LP problem; the root is the LP_1 problem.

Definition 7.1: branching node index

For each branching node k , the associated LP problem is denoted by LP_k . If LP_k has an optimal solution, it is denoted by $\tilde{\mathbf{x}}(\text{LP}_k)$ and its optimal value by $z(\text{LP}_k)$. The integer k is called the **branching node index**: it identifies the node uniquely, and the nodes are numbered in the order in which their LP problems are solved.

- A B&B algorithm keeps the best feasible solution of the problem (7.1) found during the search. This solution is called the **incumbent solution** and is denoted by $\bar{\mathbf{x}}$; its value is a **lower bound** LB on $z(\text{ILP})$. Before any feasible solution is found we set $\text{LB} = -\infty$.
- The lower bound and the incumbent solution are updated whenever a better feasible solution is found. At termination we have an **algorithmic certificate** that

$$z(\text{ILP}) = \text{LB} \quad \text{and} \quad \tilde{\mathbf{x}}(\text{ILP}) = \bar{\mathbf{x}}. \quad (7.2)$$

7.2.3 Branching operations

- The most common branching operation is the **binary branching**: it acts on a single **branching variable** and produces two child branching nodes, one with an additional **upper bound** and one with an additional **lower bound** on the branching variable.
- More precisely, let $\tilde{\mathbf{x}}(\text{LP}_k)$ be an optimal solution of the LP_k problem, and let x_j , with $j \in \{1, 2, \dots, p\}$, be an integer variable that takes a fractional value $\tilde{x}_j$. The binary branching on x_j generates two child LP problems:

1. the LP_k problem with the additional **down-branching constraint**

$$x_j \leq \lfloor \tilde{x}_j \rfloor; \quad (7.3)$$

2. the LP_k problem with the additional **up-branching constraint**

$$x_j \geq \lceil \tilde{x}_j \rceil + 1. \quad (7.4)$$

- The two child nodes are called the **down-branching node** and the **up-branching node**. In both of them $\tilde{\mathbf{x}}(\text{LP}_k)$ is no longer feasible.

Every feasible solution of the ILP problem contained in the region of LP_k satisfies either (7.3) or (7.4), so it is contained in the region of one of the two children. **This ensures the correctness of the B&B algorithm.**

7.2.4 Bounding operations and the bound of a node

- The value $z(\text{LP}_k)$ is an upper bound on the value of every feasible solution of the ILP problem in the region of LP_k . When all the values are integer, this bound can be rounded down.

Observation 7.2: rounding the bound of a node

Since all the variables and all the costs are integer, every feasible solution in the region of LP_k has value at most $\lfloor z(\text{LP}_k) \rfloor$.

Proof. Since $c_j \in \mathbb{Z}$ for all j , the value $\sum_{j=1}^n c_j \tilde{x}_j$ of a feasible solution $\tilde{\mathbf{x}}$ is a sum of products of integers, hence an integer. Since it is at most $z(\text{LP}_k)$, it is at most the largest integer not exceeding $z(\text{LP}_k)$, that is, $\lfloor z(\text{LP}_k) \rfloor$. $\square$

- Accordingly, we call **bound of the node k** the value

$$\text{UB}(\text{LP}_k) = \begin{cases} \lfloor z(\text{LP}_k) \rfloor & \text{if } p = n \text{ and } c_j \in \mathbb{Z} \text{ for all } j \in \{1, 2, \dots, n\}, \\ z(\text{LP}_k) & \text{otherwise.} \end{cases} \quad (7.5)$$

- Given a branching node k , no branching operation is performed on it if one of the following **pruning criteria** applies. In these cases the node is said to be **pruned**.

1. **Pruning by infeasibility**: the LP_k problem is infeasible, so its region contains no feasible solution of the ILP problem.

2. **Pruning by integrality:** the integer variables have integer values in $\tilde{\mathbf{x}}(\text{LP}_k)$, so $\tilde{\mathbf{x}}(\text{LP}_k)$ is feasible for the ILP problem, and it is optimal in the region of LP_k . If $z(\text{LP}_k) > \text{LB}$, the lower bound and the incumbent solution are updated.
3. **Pruning by bound:** the bound of the node does not exceed the lower bound, $\text{UB}(\text{LP}_k) \leq \text{LB}$, so the region of LP_k contains no solution better than the incumbent one.

The criteria are checked in this order: infeasibility, integrality, bound. When both integrality and bound apply, the node is pruned by integrality.

7.2.5 The list of LP problems

- A branching node that is not pruned is inserted into a **list** L of LP problems waiting for a branching operation.
- The order in which the LP problems are extracted from the list determines the **branching-node selection strategy**; the choice of the branching variable determines the **branching-variable selection strategy**. Both are discussed after the algorithm.

7.2.6 Upper bound and optimality gap

- If the list L is not empty, every feasible solution better than the incumbent one lies in the region of some node of the list. Accordingly, the B&B algorithm has the **upper bound**

$$z(\text{ILP}) \leq \text{UB} = \max_{k \in L} \{\text{UB}(\text{LP}_k)\}. \quad (7.6)$$

When the list is empty, the incumbent solution is optimal and $\text{UB} = \text{LB}$.

- The upper bound is computed after the solution of the root and updated at the end of every iteration. The algorithm does not need it, but it measures the progress through the **optimality gap**:

$$\text{optimality gap} = \frac{\text{UB} - \text{LB}}{|\text{LB}|}, \quad \text{for } \text{LB} \neq 0. \quad (7.7)$$

7.2.7 Heuristic initialization

Before the B&B algorithm, a **heuristic algorithm** can be run to find a good feasible solution. The lower bound LB and the incumbent solution $\tilde{\mathbf{x}}$ are then initialized with this solution.

- A good initial lower bound allows the algorithm to prune nodes by bound earlier, and it can reduce the size of the branch-and-bound tree considerably.
- If the initial lower bound already satisfies $\text{UB}(\text{LP}_1) \leq \text{LB}$, the root is pruned by bound: the heuristic solution is optimal and no branching is needed.

7.3 The algorithm

7.3.1 Pseudocode

Algorithm

Input: An ILP problem in canonical maximization form with integer input data.

Output: An optimal solution to the ILP problem (if one exists) or a certificate of infeasibility/unboundedness.

Step 0: Initialization

- (a) Relax the integrality constraints and solve the LP_1 problem.
 1. if $\tilde{\mathbf{x}}(LP_1)$ is **integer**, then it is an optimal solution and $z(LP_1)$ is the optimal value of the ILP problem. **STOP**.
 2. if the LP_1 problem is **infeasible**, then the ILP problem is infeasible. **STOP**.
 3. if the LP_1 problem is **unbounded**, then the ILP problem is either infeasible or unbounded. **STOP**.
- (b) Insert the LP_1 problem into a **list of LP problems for branching** (denote L). Set the **lower bound** $LB \leftarrow -\infty$ (or the value of a heuristic solution), and set a branching node counter $c \leftarrow 1$.

Step 1: Optimality check

- (a) if the list L is empty and $LB \neq -\infty$, then LB is the optimal value and the **incumbent solution** $\tilde{\mathbf{x}}(ILP)$ is an optimal solution to the ILP problem. **STOP**.
- (b) if the list L is empty and $LB = -\infty$, then the ILP problem is infeasible. **STOP**.

Step 2: Branching-node selection strategy

Select and remove an LP_k problem from the list L .

Step 3: Branching-variable selection strategy

Select a variable x_j with fractional value $\tilde{x}_j$ in $\tilde{\mathbf{x}}(LP_k)$ and generate two new LP problems:

- (a) the LP_{c+1} problem $\rightsquigarrow$ the LP_k problem with the additional **down-branching constraint**: $x_j \leq \lfloor \tilde{x}_j \rfloor$
- (b) the LP_{c+2} problem $\rightsquigarrow$ the LP_k problem with the additional **up-branching constraint**: $x_j \geq \lfloor \tilde{x}_j \rfloor + 1$

Execute the **Branching Node Solution Sub-Procedure** for the LP_{c+1} problem and for the LP_{c+2} problem; the second one is skipped if, after the first, $\lfloor z(LP_k) \rfloor \leq LB$. Increment the branching node counter $c \leftarrow c + 2$ and go to step 1.

- This is the pseudocode of the **Branching Node Solution Sub-Procedure**.

Algorithm

Input: An LP_k problem associated with a branching node.

Step 1: **Bounding operations**

Solve the LP_k problem.

- (a) if the LP_k problem is infeasible, then **STOP**.
- (b) if $\lfloor z(LP_k) \rfloor \leq LB$, then **STOP**.
- (c) if $\tilde{x}(LP_k)$ is integer, then update the **lower bound** and the **incumbent solution**:

$$LB \leftarrow z(LP_k) \quad \text{and} \quad \bar{x}(ILP) \leftarrow \tilde{x}(LP_k)$$

Remove from the list L all LP problems with $\lfloor z(LP) \rfloor \leq LB$ and **STOP**.

Step 2: **Insertion into the list**

Insert the LP_k problem into the list L and **STOP**.

- The LP problem with the down-branching constraint is called the *down-branching node*. The LP problem with the up-branching constraint is called the *up-branching node*. The order in which they are solved determines the order in which the LP problems enter the list L .
- The pseudocode does not say how the LP problems are solved: any method for LP problems can be used.
- If the **dual simplex algorithm** is used, each child is solved starting from the optimal dictionary of its parent: after adding the branching constraint this dictionary is still dual feasible, so in general a few pivot operations are enough (Section 7.4).

7.3.2 The skip rule and the numbering of the nodes

- The bound of a child never exceeds the bound of its parent, since the child has one more constraint. Accordingly, if after the solution of the first child we have $UB(LP_k) \leq LB$, the second child cannot contain a better solution, and it is **not solved**. This is the **skip rule**.
- A skipped child does not receive a node index: the nodes are numbered in the order in which their LP problems are solved, so the indices of the solved nodes are always $1, 2, \dots, c$.
- The order in which the two children are solved also determines the order in which they enter the list L . In these notes the down-branching node is solved first, unless otherwise stated.

7.3.3 Number of branching nodes

- A **complete** binary branch-and-bound tree with $n \geq 1$ levels has 2^{n-1} leaf branching nodes, and along every root-to-leaf path exactly $n - 1$ branching constraints are imposed.

Observation 7.3

The total number of branching nodes in a complete binary branch-and-bound tree with $n \geq 1$ levels is:

$$\sum_{j=0}^{n-1} 2^j = 2^n - 1. \quad (7.8)$$

Proof. By induction on n .

- **Base case.** For $n = 1$ the statement reads $\sum_{j=0}^0 2^j = 2^1 - 1$, that is, $1 = 1$, which is true.
- **Inductive step.** Suppose that the statement holds for n ; we prove it for $n + 1$. We have:

$$\begin{aligned} \sum_{j=0}^n 2^j &= \sum_{j=0}^{n-1} 2^j + 2^n = 2^n - 1 + 2^n \quad (\text{by the inductive hypothesis}) \\ &= 2^{n+1} - 1, \end{aligned}$$

which is the statement for $n + 1$. □

- The total number of nodes is the sum of the first n terms of a geometric progression with ratio $r = 2$:

$$\sum_{j=0}^{n-1} r^j = \begin{cases} \frac{r^n - 1}{r - 1} & \text{if } r \neq 1, \\ n & \text{otherwise.} \end{cases} \quad (7.9)$$

- With p binary variables the complete tree has $p + 1$ levels and $2^{p+1} - 1$ nodes: the pruning criteria are what makes the algorithm practical.

7.4 A short recap of the dual simplex algorithm

- A child differs from its parent by one constraint. Solving it from scratch would waste the work done for the parent: the B&B algorithm starts instead from the **optimal dictionary of the parent** and re-optimizes it with the **dual simplex algorithm**.
- Let x_j be the branching variable. It is basic in the optimal dictionary of LP_k , since its value is fractional, and its row reads

$$x_j = \bar{b}_j - \sum_{q \in \mathcal{N}} \bar{a}_{jq} x_q, \quad \text{with } \bar{b}_j = \tilde{x}_j.$$

- At depth d of the tree, the branching constraint receives the slack (or surplus) variable x_s , with $s = n + m + d$.

Observation 7.4: the row of the branching constraint

Substituting the row of x_j , the branching constraints become new basic rows of the dictionary:

$$x_j \leq \lfloor \tilde{x}_j \rfloor \implies x_s = \lfloor \tilde{x}_j \rfloor - x_j = (\lfloor \tilde{x}_j \rfloor - \tilde{x}_j) + \sum_{q \in \mathcal{N}} \bar{a}_{jq} x_q,$$

$$x_j \geq \lfloor \tilde{x}_j \rfloor + 1 \implies x_s = x_j - \lfloor \tilde{x}_j \rfloor - 1 = (\tilde{x}_j - \lfloor \tilde{x}_j \rfloor - 1) - \sum_{q \in \mathcal{N}} \bar{a}_{jq} x_q.$$

In both cases the value of x_s is negative, while the reduced costs do not change: the new dictionary is **dual feasible** but not primal feasible.

- This is exactly the starting point of the dual simplex algorithm. In these notes we use the same pivoting rules as in the module *Linear Programming*.

Dual pivoting operation on a dictionary $x_i = \bar{b}_i - \sum_{j \in \mathcal{N}} \bar{a}_{ij} x_j$, for $i \in \mathcal{B}$, and $z = \bar{z} + \sum_{j \in \mathcal{N}} \bar{c}_j x_j$, with $\bar{c}_j \leq 0$ for all $j \in \mathcal{N}$:

- a. the **pivot row** is the basic variable with negative value having the smallest index:

$$r = \min \{ i \in \mathcal{B} : \bar{b}_i < 0 \};$$

- b. if $\bar{a}_{rj} \geq 0$ for all $j \in \mathcal{N}$, the LP problem is **infeasible**;
- c. otherwise the **pivot column** x_p attains the minimum ratio (ties: smallest index)

$$\frac{\bar{c}_p}{\bar{a}_{rp}} = s_m = \min \left\{ \frac{\bar{c}_j}{\bar{a}_{rj}} : j \in \mathcal{N}, \bar{a}_{rj} < 0 \right\};$$

- d. after the pivoting operation the objective value is $\bar{z} + s_m \bar{b}_r \leq \bar{z}$.

The algorithm stops when all the basic variables are non-negative: the dictionary is then optimal.

- In a child node one or two dual pivoting operations are often enough, while the simplex algorithm from the slack basis would need several pivoting operations for every node.

7.5 Branching-variable selection strategies

- The choice of the branching variable has a large impact on the size of the branch-and-bound tree.

Ideally, the branching variable gives the largest decrease of the bounds of the children and leads quickly to feasible solutions. However, the time spent to choose it also counts: an expensive rule may slow down the whole algorithm.

- In all the strategies below, ties are broken by the **smallest index**.

7.5.1 Lexicographic

The **lexicographic** strategy selects the integer variable with a fractional value having the smallest (or the largest) index.

- It requires no computation, but it takes into account neither the decrease of the bounds nor the search for feasible solutions. It is the rule used in the exercises, because it makes the execution easy to reproduce.

7.5.2 Minimum and maximum infeasibility

The **minimum** (respectively, **maximum**) **infeasibility** strategy selects the variable whose fractional value is the closest to (respectively, the farthest from) an integer value.

- Given a branching node k , for each integer variable x_j with a fractional value $\tilde{x}_j$ in $\tilde{\mathbf{x}}(\text{LP}_k)$, the **fractional value** of the variable is

$$f_j = \min \{ \tilde{x}_j - \lfloor \tilde{x}_j \rfloor, \lceil \tilde{x}_j \rceil - \tilde{x}_j \}. \quad (7.10)$$

- Minimum infeasibility tries to reduce the number of fractional variables in the children, to reach feasible solutions faster.
- Maximum infeasibility removes first the variables that contribute more to the infeasibility of $\tilde{\mathbf{x}}(\text{LP}_k)$. Neither rule looks at the bounds of the children.

Example 7.1: fractional values

At the root node of the car example we have $\tilde{x}_1 = \frac{7}{2}$, $\tilde{x}_2 = \frac{5}{2}$, and the fractional values are:

$$f_1 = \min \left\{ \frac{7}{2} - 3, 4 - \frac{7}{2} \right\} = \frac{1}{2} \quad f_2 = \min \left\{ \frac{5}{2} - 2, 3 - \frac{5}{2} \right\} = \frac{1}{2}$$

The two values are equal: both minimum and maximum infeasibility have a tie, which is broken by the smallest index, so both strategies select x_1 .

7.5.3 Strong branching

The **strong branching** strategy selects the variable that gives the largest decrease of the bounds of the children.

- Given a branching node k , for each integer variable x_j with a fractional value, we solve the two children and compute the decreases of the optimal value

$$\delta_k^d(x_j) = z(\text{LP}_k) - z(\text{down-branching child}), \quad \delta_k^u(x_j) = z(\text{LP}_k) - z(\text{up-branching child}),$$

where the decrease is $+\infty$ for an infeasible child.

- The branching variable is x_s , with

$$s = \arg \max \left\{ \min \left\{ \delta_k^d(x_j), \delta_k^u(x_j) \right\} : j \in \{1, 2, \dots, p\}, \tilde{x}_j \text{ fractional} \right\}. \quad (7.11)$$

Both children of the selected variable have a large decrease, so both are more likely to be pruned by bound.

- Strong branching is expensive, since it solves two LP problems for every candidate variable. For this reason it is typically applied only at the top levels of the tree.

Example 7.2: strong branching

At the root node of the car example we have $\tilde{x}_1 = \frac{7}{2}$, $\tilde{x}_2 = \frac{5}{2}$ and $z(\text{LP}_1) = \frac{31}{2}$. For each fractional variable we solve the two child LP problems (with the dual simplex algorithm, from the optimal dictionary of LP_1):

Variable	Down	z	$\delta_1^d(x_j)$	Up	z	$\delta_1^u(x_j)$	$\min\{\delta_1^d, \delta_1^u\}$
x_1	$x_1 \leq 3$	$\frac{100}{7}$	$\frac{17}{14}$	$x_1 \geq 4$	$\frac{29}{2}$	1	1
x_2	$x_2 \leq 2$	$\frac{151}{10}$	$\frac{2}{5}$	$x_2 \geq 3$	$\frac{45}{4}$	$\frac{17}{4}$	$\frac{2}{5}$

The largest minimum decrease is 1: strong branching selects x_1 , the same variable chosen by the smallest-index rule.

7.5.4 Further strategies

- Solvers estimate the decreases δ_k^d and δ_k^u from the history of the search, without solving the children, by means of **pseudocosts**; the same estimates give a branching-node selection rule called **best estimate**. These strategies are a possible extension of this chapter.

7.6 Branching-node selection strategies

- The order in which the nodes of the list L are processed is the **branching-node selection strategy**: it decides where the search goes next.

7.6.1 Best bound

The **best-bound** strategy selects the node of the list L with the largest optimal value $z(LP_k)$. Ties are broken by the smallest node index.

- The selected node gives the upper bound (7.6): processing it is the only way to decrease UB. The rule tends to explore few nodes, but the list can become long.

7.6.2 LIFO and FIFO

The **Last-In-First-Out** (LIFO) strategy processes the nodes in the reverse order of insertion into the list L . It is equivalent to a **depth-first search** (DFS).

- The list is implemented as a **stack**. The search goes deep into the tree, where the leaves often contain feasible solutions: the goal is to update the lower bound and the incumbent solution quickly. The list stays short.

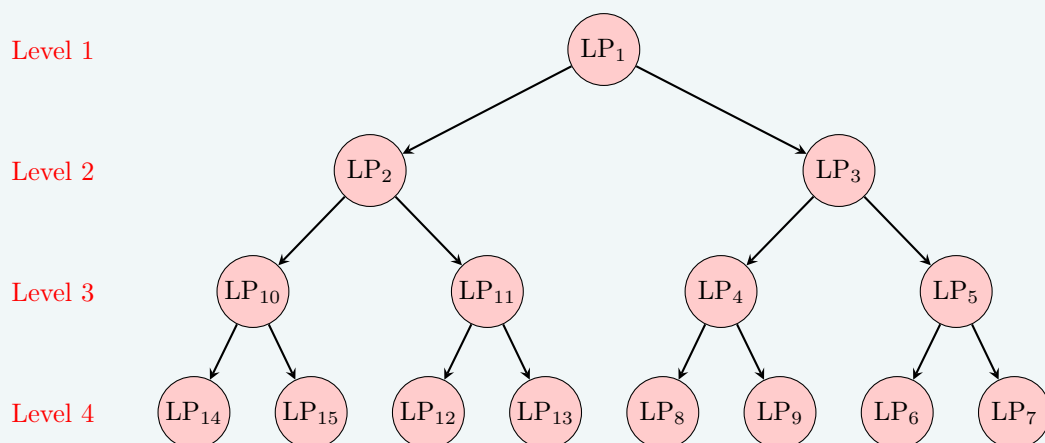
The **First-In-First-Out** (FIFO) strategy processes the nodes in the order of insertion into the list L . It is equivalent to a **breadth-first search** (BFS).

- The list is implemented as a **queue**. The tree is explored level by level: this improves the upper bound, but usually it does not find good lower bounds quickly.

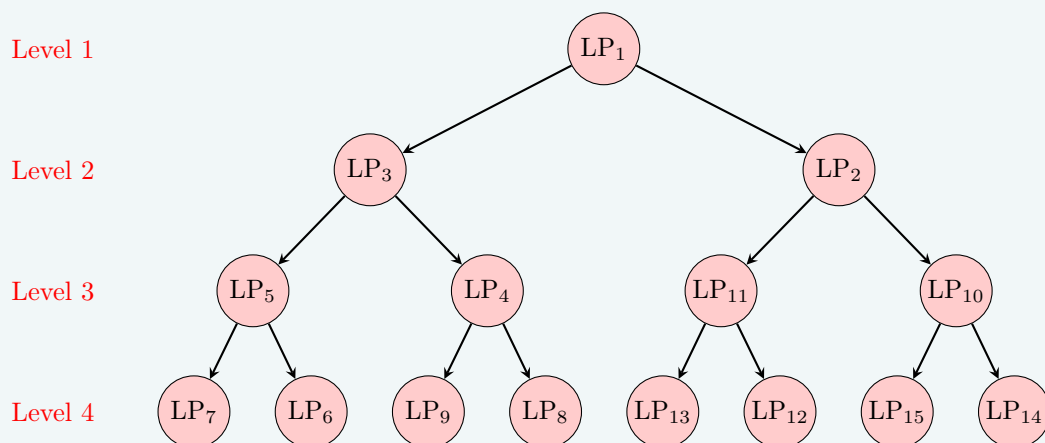
Example 7.3: LIFO strategy

Order in which a complete binary branch-and-bound tree of 4 levels is explored with the LIFO strategy (no node is pruned).

1. If the down-branching node is solved first, we have:



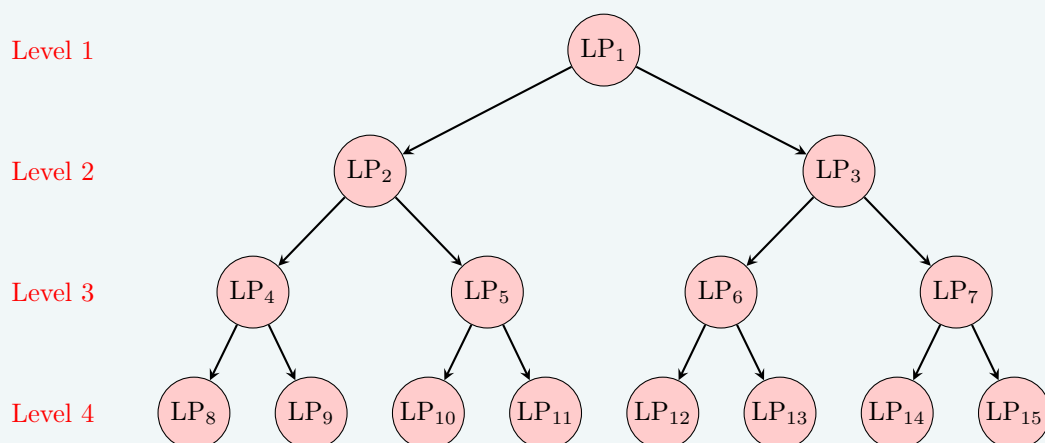
2. If the up-branching node is solved first, we have:



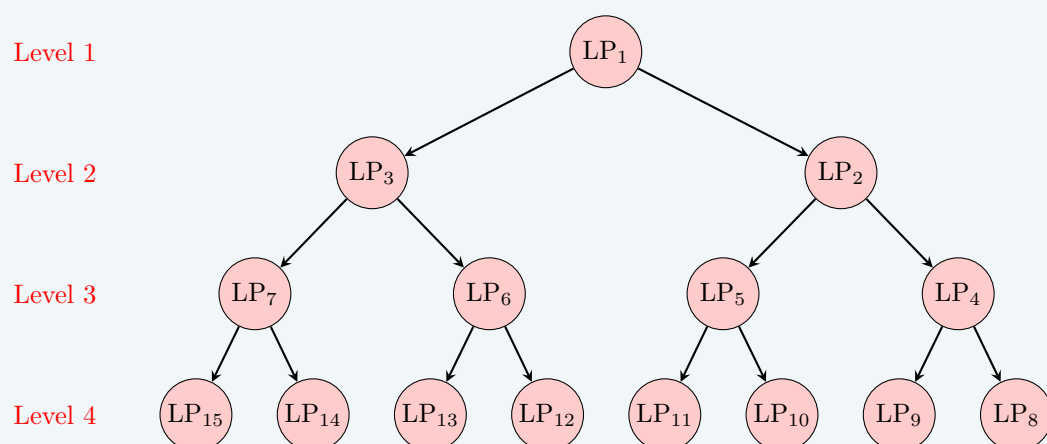
Example 7.4: FIFO strategy

Order in which a complete binary branch-and-bound tree of 4 levels is explored with the FIFO strategy (no node is pruned).

1. If the down-branching node is solved first, we have:



2. If the up-branching node is solved first, we have:



7.7 Worked examples on the car example

- We now solve the car example with two executions of the B&B algorithm, which differ only in the selection strategies. Since all the variables and all the costs are integer, the bound of a node is $UB(LP_k) = \lfloor z(LP_k) \rfloor$.
- In the first execution every node is re-optimized with the dual simplex algorithm, and we report all the dictionaries. For each node a figure shows its feasible region.
- Before reading the solution, try to predict which variable is selected at the root, and which node is extracted at the second iteration.

Example 7.5: execution A: smallest index and LIFO

We execute the B&B algorithm on the car example with:

1. the smallest-index rule (lexicographic) as branching-variable selection strategy;
2. the LIFO rule as branching-node selection strategy;
3. the down-branching node is solved first, then the up-branching node;
4. the second child is not solved when the skip rule applies;
5. the lower bound is initialized to $LB = -\infty$ (no heuristic algorithm is executed).

• Initialization

The LP_1 problem is the LP relaxation of the car example. The simplex algorithm with Bland's rule, starting from the basis of the slack variables, performs 2 pivoting operations (x_1 enters and x_3 leaves; x_2 enters and x_4 leaves) and ends with the optimal dictionary:

$$\begin{cases} x_1 = \frac{7}{2} - \frac{7}{62}x_3 + \frac{1}{31}x_4 \\ x_2 = \frac{5}{2} + \frac{1}{31}x_3 - \frac{5}{62}x_4 \\ z = \frac{31}{2} - \frac{17}{62}x_3 - \frac{2}{31}x_4 \end{cases}$$

An optimal solution $\tilde{\mathbf{x}}(\text{LP}_1)$ is:

$$\tilde{x}_1 = \frac{7}{2}, \quad \tilde{x}_2 = \frac{5}{2}, \quad z(\text{LP}_1) = \frac{31}{2}.$$

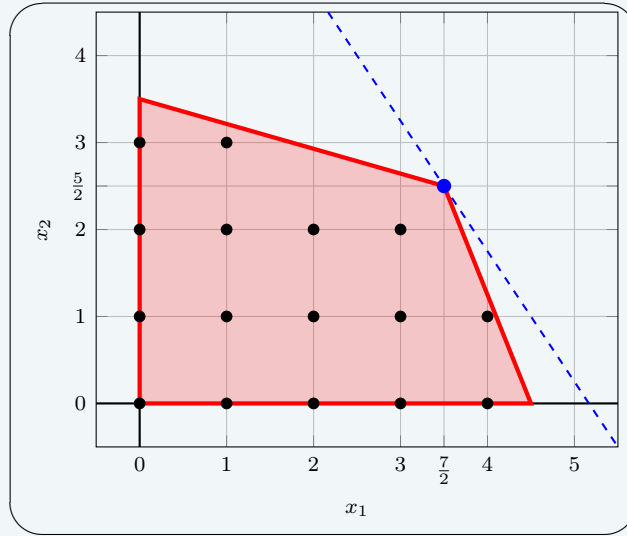


Figure 7.2: Node 1: the feasible region of LP_1 , the integer points, the optimal solution of LP_1 (blue) and the objective line through it (dashed).

The solution is fractional and $\lfloor z(\text{LP}_1) \rfloor = \left\lfloor \frac{31}{2} \right\rfloor = 15 > \text{LB} = -\infty$: the LP_1 problem is inserted in the list L .

The list is $L = \{\text{LP}_1\}$, and the upper bound is the largest bound of the nodes in the list:

$$\underbrace{-\infty}_{\text{LB}} \leq z(\text{ILP}) \leq \underbrace{15}_{\text{UB}} \quad \left(\text{UB} = \max_{k \in L} \text{UB}(\text{LP}_k) = 15 \right).$$

- **Iteration 1**

The list is $L = \{\text{LP}_1\}$: the LP_1 problem is extracted and removed from L .

In $\tilde{\mathbf{x}}(\text{LP}_1)$ we have $\tilde{x}_1 = \frac{7}{2}$ and $\tilde{x}_2 = \frac{5}{2}$, both fractional; by the smallest-index rule the branching variable is x_1 . The branching constraints are:

$$x_1 \leq \underbrace{\left\lfloor \frac{7}{2} \right\rfloor}_{=3} \quad \text{and} \quad x_1 \geq \underbrace{\left\lceil \frac{7}{2} \right\rceil}_{=4} + 1.$$

1. **Node 2: the down-branching child** ($x_1 \leq 3$).

The branching constraint $x_1 \leq 3$ is written as $x_1 + x_5 = 3$, with the slack variable $x_5 \geq 0$. Using the row of x_1 in the optimal dictionary of LP_1 , we have:

$$x_5 = 3 - x_1 = 3 - \left(\frac{7}{2} - \frac{7}{62}x_3 + \frac{1}{31}x_4 \right) = -\frac{1}{2} + \frac{7}{62}x_3 - \frac{1}{31}x_4.$$

We add this row to the optimal dictionary of LP_1 . The reduced costs do not change, so the new dictionary is **dual feasible**; it is not primal feasible, since $\bar{b}_5 = -\frac{1}{2} < 0$:

$$\left\{ \begin{array}{rclcl} x_1 & = & \frac{7}{2} & - & \frac{7}{62}x_3 & + & \frac{1}{31}x_4 \\ x_2 & = & \frac{5}{2} & + & \frac{1}{31}x_3 & - & \frac{5}{62}x_4 \\ x_5 & = & -\frac{1}{2} & + & \frac{7}{62}x_3 & - & \frac{1}{31}x_4 \\ z & = & \frac{31}{2} & - & \frac{17}{62}x_3 & - & \frac{2}{31}x_4 \end{array} \right.$$

Dual pivoting operation. The pivot row is x_5 , the only basic variable with a negative value ($\bar{b}_5 = -\frac{1}{2}$).

The candidates for the pivot column are the non-basic variables with $\bar{a}_{5,j} < 0$:

$$\frac{\bar{c}_3}{\bar{a}_{5,3}} = \frac{-\frac{17}{62}}{-\frac{7}{62}} = \frac{17}{7}$$

The pivot column is x_3 , with $s_m = \frac{17}{7}$, and the objective value becomes

$$\bar{z} + s_m \bar{b}_5 = \frac{31}{2} + \frac{17}{7} \cdot \left(-\frac{1}{2} \right) = \frac{100}{7}.$$

After the pivoting operation we obtain the following dictionary, which is primal and dual feasible, hence **optimal**:

$$\left\{ \begin{array}{rclcl} x_1 & = & 3 & & - & x_5 \\ x_2 & = & \frac{37}{14} & - & \frac{1}{14}x_4 & + & \frac{2}{7}x_5 \\ x_3 & = & \frac{31}{7} & + & \frac{2}{7}x_4 & + & \frac{62}{7}x_5 \\ z & = & \frac{100}{7} & - & \frac{1}{7}x_4 & - & \frac{17}{7}x_5 \end{array} \right.$$

An optimal solution $\tilde{\mathbf{x}}(LP_2)$ is:

$$\tilde{x}_1 = 3, \quad \tilde{x}_2 = \frac{37}{14}, \quad z(LP_2) = \frac{100}{7}.$$

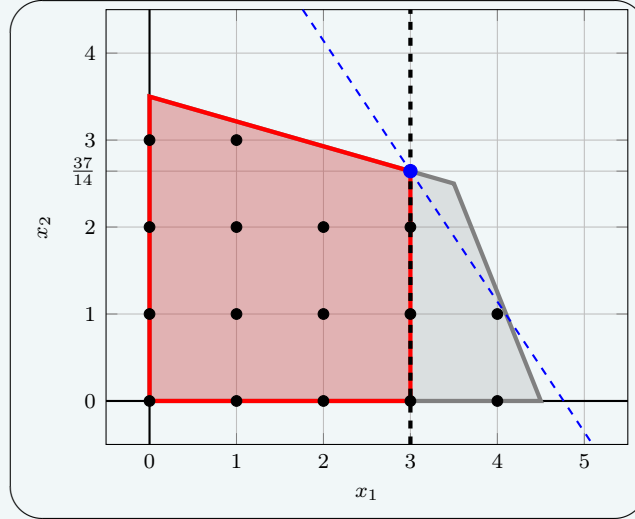


Figure 7.3: Node 2: the feasible region of LP_2 (red) inside the region of its parent LP_1 (gray), and the optimal solution of LP_2 (blue) with the objective line through it (dashed).

The solution is **fractional** and

$$\lfloor z(LP_2) \rfloor = \left\lfloor \frac{100}{7} \right\rfloor = 14 > LB = -\infty,$$

so the LP_2 problem is **inserted in the list L** .

2. **Node 3: the up-branching child ($x_1 \geq 4$).**

The branching constraint $x_1 \geq 4$ is written as $x_1 - x_5 = 4$, with the surplus variable $x_5 \geq 0$. Using the row of x_1 in the optimal dictionary of LP_1 , we have:

$$x_5 = x_1 - 4 = \left(\frac{7}{2} - \frac{7}{62}x_3 + \frac{1}{31}x_4 \right) - 4 = -\frac{1}{2} - \frac{7}{62}x_3 + \frac{1}{31}x_4.$$

We add this row to the optimal dictionary of LP_1 . The reduced costs do not change, so the new dictionary is **dual feasible**; it is not primal feasible, since $\bar{b}_5 = -\frac{1}{2} < 0$:

$$\begin{cases} x_1 &= \frac{7}{2} - \frac{7}{62}x_3 + \frac{1}{31}x_4 \\ x_2 &= \frac{5}{2} + \frac{1}{31}x_3 - \frac{5}{62}x_4 \\ x_5 &= -\frac{1}{2} - \frac{7}{62}x_3 + \frac{1}{31}x_4 \\ z &= \frac{31}{2} - \frac{17}{62}x_3 - \frac{2}{31}x_4 \end{cases}$$

Dual pivoting operation. The pivot row is x_5 , the only basic variable with a negative value ($\bar{b}_5 = -\frac{1}{2}$).

The candidates for the pivot column are the non-basic variables with $\bar{a}_{5,j} < 0$:

$$\frac{\bar{c}_4}{\bar{a}_{5,4}} = \frac{-\frac{2}{31}}{-\frac{1}{31}} = 2$$

The pivot column is x_4 , with $s_m = 2$, and the objective value becomes

$$\bar{z} + s_m \bar{b}_5 = \frac{31}{2} + 2 \cdot \left(-\frac{1}{2}\right) = \frac{29}{2}.$$

After the pivoting operation we obtain the following dictionary, which is primal and dual feasible, hence **optimal**:

$$\begin{cases} x_1 &= 4 & & + & x_5 \\ x_2 &= \frac{5}{4} - \frac{1}{4}x_3 - \frac{5}{2}x_5 \\ x_4 &= \frac{31}{2} + \frac{7}{2}x_3 + 31x_5 \\ z &= \frac{29}{2} - \frac{1}{2}x_3 - 2x_5 \end{cases}$$

An optimal solution $\tilde{\mathbf{x}}(\text{LP}_3)$ is:

$$\tilde{x}_1 = 4, \quad \tilde{x}_2 = \frac{5}{4}, \quad z(\text{LP}_3) = \frac{29}{2}.$$

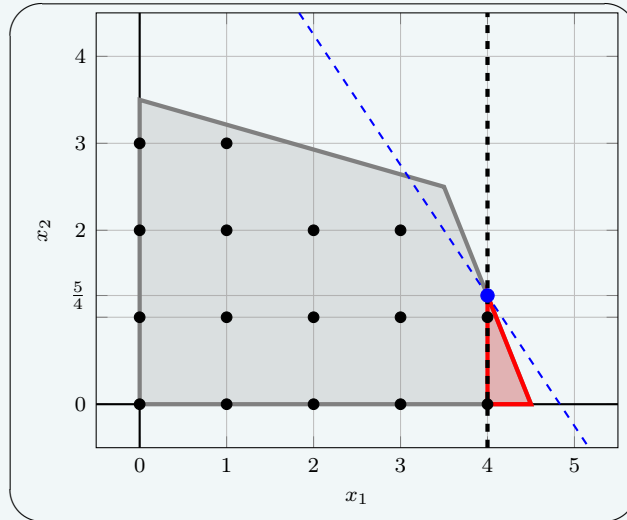


Figure 7.4: Node 3: the feasible region of LP_3 (red) inside the region of its parent LP_1 (gray), and the optimal solution of LP_3 (blue) with the objective line through it (dashed).

The solution is **fractional** and

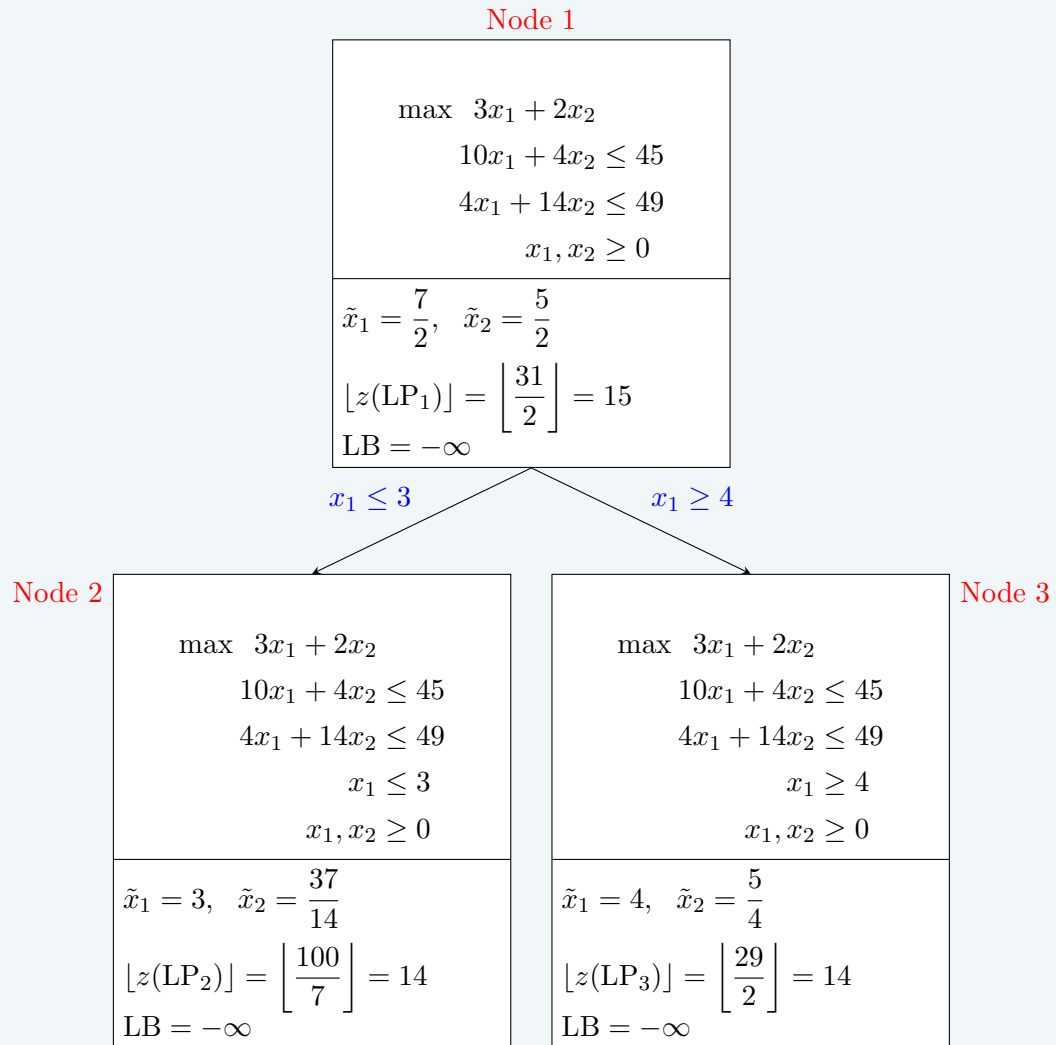
$$\lfloor z(\text{LP}_3) \rfloor = \left\lfloor \frac{29}{2} \right\rfloor = 14 > \text{LB} = -\infty,$$

so the LP_3 problem is **inserted in the list** L .

The list is $L = \{LP_2, LP_3\}$, and the upper bound is the largest bound of the nodes in the list:

$$\underbrace{-\infty}_{\text{LB}} \leq z(\text{ILP}) \leq \underbrace{14}_{\text{UB}} \quad \left(\text{UB} = \max_{k \in L} \text{UB}(LP_k) = 14 \right).$$

This is the portion of the branch-and-bound tree built in the first iteration, with the LP problems of the branching nodes:



• **Iteration 2**

The list is $L = \{LP_2, LP_3\}$; following the LIFO rule, LP_3 (the last problem inserted) is extracted and removed from L .

In $\tilde{\mathbf{x}}(LP_3)$ the only integer variable with a fractional value is x_2 , which is the branching variable. The branching constraints are:

$$x_2 \leq \underbrace{\left\lfloor \frac{5}{4} \right\rfloor}_{=1} \quad \text{and} \quad x_2 \geq \underbrace{\left\lfloor \frac{5}{4} \right\rfloor + 1}_{=2}.$$

1. **Node 4: the down-branching child** ($x_2 \leq 1$).

The branching constraint $x_2 \leq 1$ is written as $x_2 + x_6 = 1$, with the slack variable $x_6 \geq 0$. Using the row of x_2 in the optimal dictionary of LP_3 , we have:

$$x_6 = 1 - x_2 = 1 - \left(\frac{5}{4} - \frac{1}{4}x_3 - \frac{5}{2}x_5 \right) = -\frac{1}{4} + \frac{1}{4}x_3 + \frac{5}{2}x_5.$$

We add this row to the optimal dictionary of LP_3 . The reduced costs do not change, so the new dictionary is **dual feasible**; it is not primal feasible, since $\bar{b}_6 = -\frac{1}{4} < 0$:

$$\left\{ \begin{array}{rclcl} x_1 & = & 4 & + & x_5 \\ x_2 & = & \frac{5}{4} & - & \frac{1}{4}x_3 - \frac{5}{2}x_5 \\ x_4 & = & \frac{31}{2} & + & \frac{7}{2}x_3 + 31x_5 \\ x_6 & = & -\frac{1}{4} & + & \frac{1}{4}x_3 + \frac{5}{2}x_5 \\ z & = & \frac{29}{2} & - & \frac{1}{2}x_3 - 2x_5 \end{array} \right.$$

Dual pivoting operation. The pivot row is x_6 , the only basic variable with a negative value ($\bar{b}_6 = -\frac{1}{4}$).

The candidates for the pivot column are the non-basic variables with $\bar{a}_{6,j} < 0$:

$$\frac{\bar{c}_3}{\bar{a}_{6,3}} = \frac{-\frac{1}{2}}{-\frac{1}{4}} = 2, \quad \frac{\bar{c}_5}{\bar{a}_{6,5}} = \frac{-2}{-\frac{5}{2}} = \frac{4}{5}$$

$$s_m = \min \left\{ 2, \frac{4}{5} \right\} = \frac{4}{5}.$$

The pivot column is x_5 , with $s_m = \frac{4}{5}$, and the objective value becomes

$$\bar{z} + s_m \bar{b}_6 = \frac{29}{2} + \frac{4}{5} \cdot \left(-\frac{1}{4} \right) = \frac{143}{10}.$$

After the pivoting operation we obtain the following dictionary, which is primal and dual feasible, hence **optimal**:

$$\left\{ \begin{array}{rclcl} x_1 & = & \frac{41}{10} & - & \frac{1}{10}x_3 + \frac{2}{5}x_6 \\ x_2 & = & 1 & & -x_6 \\ x_4 & = & \frac{93}{5} & + & \frac{2}{5}x_3 + \frac{62}{5}x_6 \\ x_5 & = & \frac{1}{10} & - & \frac{1}{10}x_3 + \frac{2}{5}x_6 \\ z & = & \frac{143}{10} & - & \frac{3}{10}x_3 - \frac{4}{5}x_6 \end{array} \right.$$

An optimal solution $\tilde{x}(\text{LP}_4)$ is:

$$\tilde{x}_1 = \frac{41}{10}, \quad \tilde{x}_2 = 1, \quad z(\text{LP}_4) = \frac{143}{10}.$$

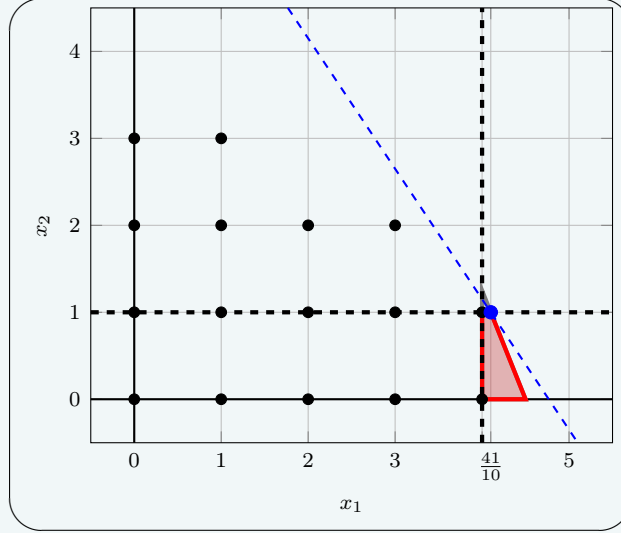


Figure 7.5: Node 4: the feasible region of LP_4 (red) inside the region of its parent LP_3 (gray), and the optimal solution of LP_4 (blue) with the objective line through it (dashed).

The solution is **fractional** and

$$\lfloor z(\text{LP}_4) \rfloor = \left\lfloor \frac{143}{10} \right\rfloor = 14 > \text{LB} = -\infty,$$

so the LP_4 problem is **inserted in the list L** .

2. **Node 5: the up-branching child ($x_2 \geq 2$).**

The branching constraint $x_2 \geq 2$ is written as $x_2 - x_6 = 2$, with the surplus variable $x_6 \geq 0$. Using the row of x_2 in the optimal dictionary of LP_3 , we have:

$$x_6 = x_2 - 2 = \left(\frac{5}{4} - \frac{1}{4}x_3 - \frac{5}{2}x_5 \right) - 2 = -\frac{3}{4} - \frac{1}{4}x_3 - \frac{5}{2}x_5.$$

We add this row to the optimal dictionary of LP_3 . The reduced costs do not change, so the new dictionary is **dual feasible**; it is not primal feasible, since $\bar{b}_6 = -\frac{3}{4} < 0$:

$$\left\{ \begin{array}{lcl} x_1 & = & 4 \quad \quad \quad + \quad x_5 \\ x_2 & = & \frac{5}{4} - \frac{1}{4}x_3 - \frac{5}{2}x_5 \\ x_4 & = & \frac{31}{2} + \frac{7}{2}x_3 + 31x_5 \\ x_6 & = & -\frac{3}{4} - \frac{1}{4}x_3 - \frac{5}{2}x_5 \\ z & = & \frac{29}{2} - \frac{1}{2}x_3 - 2x_5 \end{array} \right.$$

Dual pivoting operation. The pivot row is x_6 , the only basic variable with a negative value ($\bar{b}_6 = -\frac{3}{4}$).

All the coefficients of the pivot row are non-negative ($\bar{a}_{6,3} = \frac{1}{4}$, $\bar{a}_{6,5} = \frac{5}{2}$), that is, the row reads

$$x_6 = -\frac{3}{4} - \frac{1}{4}x_3 - \frac{5}{2}x_5 \leq -\frac{3}{4} < 0$$

for every non-negative value of the non-basic variables. By the dual simplex algorithm, the LP_5 problem is **infeasible**.

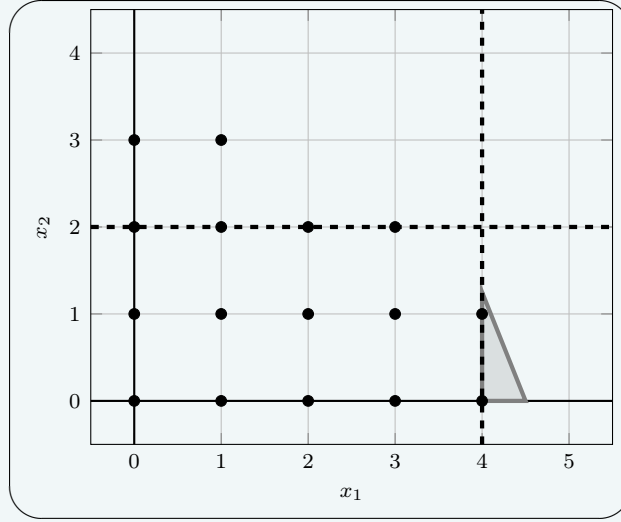


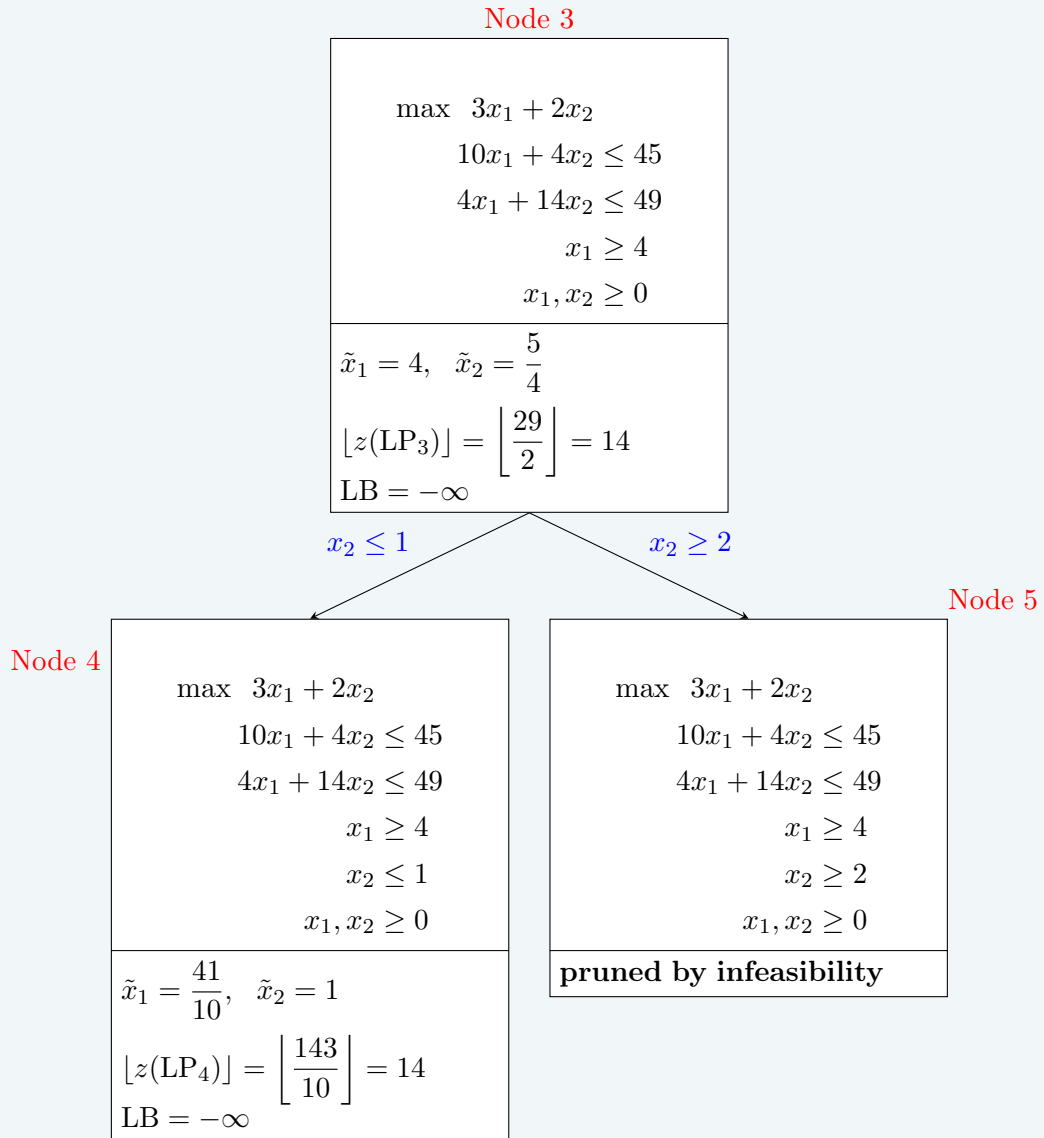
Figure 7.6: Node 5: the constraint $x_2 \geq 2$ leaves no point of the region of its parent LP_3 (gray): LP_5 is infeasible.

Accordingly, Node 5 is pruned by infeasibility.

The list is $L = \{LP_2, LP_4\}$, and the upper bound is the largest bound of the nodes in the list:

$$\underbrace{-\infty}_{\text{LB}} \leq z(\text{ILP}) \leq \underbrace{14}_{\text{UB}} \quad \left(\text{UB} = \max_{k \in L} \text{UB}(LP_k) = 14 \right).$$

This is the portion of the branch-and-bound tree built in the second iteration, with the LP problems of the branching nodes:



• **Iteration 3**

The list is $L = \{\text{LP}_2, \text{LP}_4\}$; following the LIFO rule, LP_4 (the last problem inserted) is extracted and removed from L .

In $\tilde{x}(\text{LP}_4)$ the only integer variable with a fractional value is x_1 , which is the branching variable. The branching constraints are:

$$x_1 \leq \underbrace{\left\lfloor \frac{41}{10} \right\rfloor}_{=4} \quad \text{and} \quad x_1 \geq \underbrace{\left\lfloor \frac{41}{10} \right\rfloor + 1}_{=5}.$$

1. **Node 6: the down-branching child ($x_1 \leq 4$).**

The branching constraint $x_1 \leq 4$ is written as $x_1 + x_7 = 4$, with the slack variable $x_7 \geq 0$. Using the row of x_1 in the optimal dictionary of LP_4 , we have:

$$x_7 = 4 - x_1 = 4 - \left(\frac{41}{10} - \frac{1}{10} x_3 + \frac{2}{5} x_6 \right) = -\frac{1}{10} + \frac{1}{10} x_3 - \frac{2}{5} x_6.$$

We add this row to the optimal dictionary of LP_4 . The reduced costs do not change, so the new dictionary is **dual feasible**; it is not primal feasible, since $\bar{b}_7 = -\frac{1}{10} < 0$:

$$\left\{ \begin{array}{rcllcl} x_1 & = & \frac{41}{10} & - & \frac{1}{10} x_3 & + & \frac{2}{5} x_6 \\ x_2 & = & 1 & & & - & x_6 \\ x_4 & = & \frac{93}{5} & + & \frac{2}{5} x_3 & + & \frac{62}{5} x_6 \\ x_5 & = & \frac{1}{10} & - & \frac{1}{10} x_3 & + & \frac{2}{5} x_6 \\ x_7 & = & -\frac{1}{10} & + & \frac{1}{10} x_3 & - & \frac{2}{5} x_6 \\ z & = & \frac{143}{10} & - & \frac{3}{10} x_3 & - & \frac{4}{5} x_6 \end{array} \right.$$

Dual pivoting operation. The pivot row is x_7 , the only basic variable with a negative value ($\bar{b}_7 = -\frac{1}{10}$).

The candidates for the pivot column are the non-basic variables with $\bar{a}_{7,j} < 0$:

$$\frac{\bar{c}_3}{\bar{a}_{7,3}} = \frac{-\frac{3}{10}}{-\frac{1}{10}} = 3$$

The pivot column is x_3 , with $s_m = 3$, and the objective value becomes

$$\bar{z} + s_m \bar{b}_7 = \frac{143}{10} + 3 \cdot \left(-\frac{1}{10} \right) = 14.$$

After the pivoting operation we obtain the following dictionary, which is primal and dual feasible, hence **optimal**:

$$\left\{ \begin{array}{rcllcl} x_1 & = & 4 & & - & x_7 \\ x_2 & = & 1 & - & & x_6 \\ x_3 & = & 1 & + & 4 x_6 & + & 10 x_7 \\ x_4 & = & 19 & + & 14 x_6 & + & 4 x_7 \\ x_5 & = & 0 & & - & x_7 \\ z & = & 14 & - & 2 x_6 & - & 3 x_7 \end{array} \right.$$

An optimal solution $\tilde{\mathbf{x}}(LP_6)$ is:

$$\tilde{x}_1 = 4, \quad \tilde{x}_2 = 1, \quad z(LP_6) = 14.$$

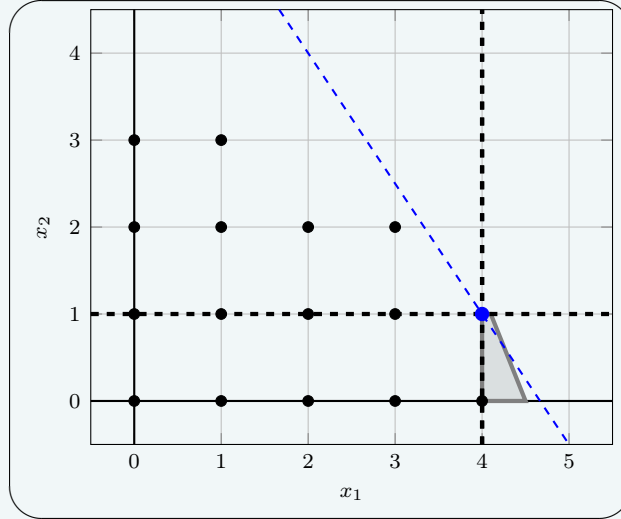


Figure 7.7: Node 6: the feasible region of LP_6 (red) inside the region of its parent LP_4 (gray), and the optimal solution of LP_6 (blue) with the objective line through it (dashed).

The solution $\tilde{\mathbf{x}}(LP_6)$ is **integer**.

Since $z(LP_6) = 14 > LB = -\infty$, the lower bound and the incumbent solution are updated:

$$LB \leftarrow 14, \quad \bar{\mathbf{x}} \leftarrow (4, 1).$$

Node 6 is pruned by integrality.

Moreover, LP_2 is removed from the list L (pruned by bound), since now

$$\lfloor z(LP_2) \rfloor = \left\lfloor \frac{100}{7} \right\rfloor = 14 \leq LB = 14.$$

2. **Skip rule.** After Node 6 the lower bound satisfies

$$\lfloor z(LP_4) \rfloor = \left\lfloor \frac{143}{10} \right\rfloor = 14 \leq LB = 14,$$

so no solution in the region of LP_4 can improve the incumbent solution: the up-branching child ($x_1 \geq 5$) is **not solved** and it does not receive a node index.

The list is empty, so the upper bound is equal to the lower bound:

$$\underbrace{14}_{LB} \leq z(ILP) \leq \underbrace{14}_{UB} \quad (UB = LB = 14).$$

This is the portion of the branch-and-bound tree built in the third iteration, with the LP problems of the branching nodes:

Node 4

$$\begin{aligned}
 \max \quad & 3x_1 + 2x_2 \\
 & 10x_1 + 4x_2 \leq 45 \\
 & 4x_1 + 14x_2 \leq 49 \\
 & x_1 \geq 4 \\
 & x_2 \leq 1 \\
 & x_1, x_2 \geq 0
 \end{aligned}$$

$$\tilde{x}_1 = \frac{41}{10}, \quad \tilde{x}_2 = 1$$

$$\lfloor z(\text{LP}_4) \rfloor = \left\lfloor \frac{143}{10} \right\rfloor = 14$$

$$\text{LB} = -\infty$$

$$x_1 \leq 4$$

Node 6

$$\begin{aligned}
 \max \quad & 3x_1 + 2x_2 \\
 & 10x_1 + 4x_2 \leq 45 \\
 & 4x_1 + 14x_2 \leq 49 \\
 & x_1 \geq 4 \\
 & x_2 \leq 1 \\
 & x_1 \leq 4 \\
 & x_1, x_2 \geq 0
 \end{aligned}$$

$$\tilde{x}_1 = 4, \quad \tilde{x}_2 = 1$$

$$z(\text{LP}_6) = 14$$

$$\text{LB} = 14$$

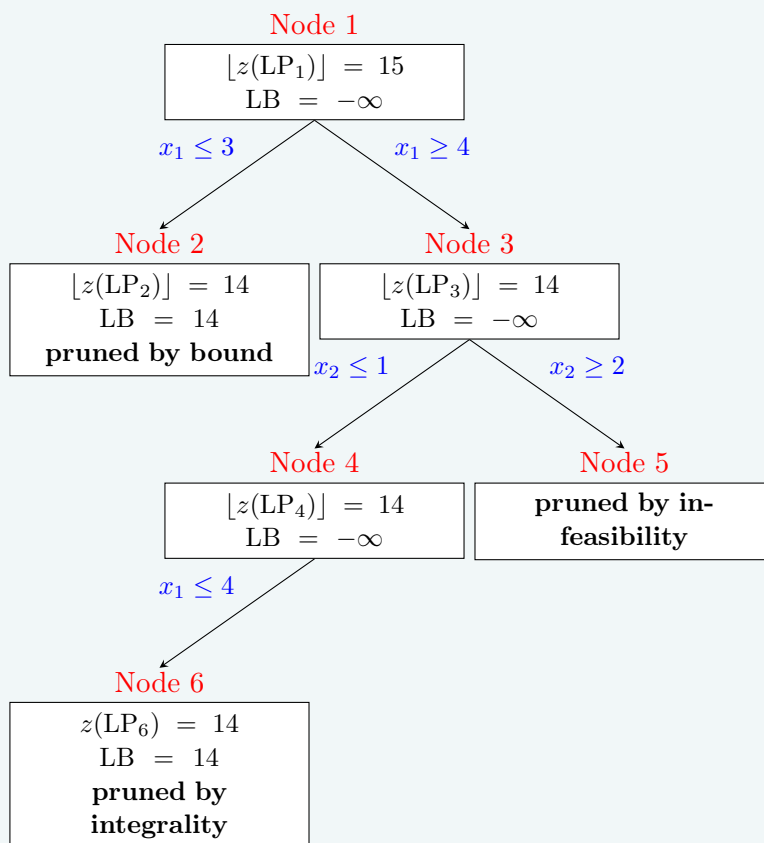
pruned by integrality

- **Termination**

The list L is empty: $\text{LB} = \text{UB} = 14$ and the incumbent solution is optimal:

$$\tilde{x}_1 = 4, \quad \tilde{x}_2 = 1, \quad z(\text{ILP}) = 14.$$

This is the final branch-and-bound tree. For each branching node it shows the bound of the node, the lower bound LB when the node was solved (or removed from the list) and the pruning criterion.



The same execution in a table:

Node	Parent	It.	Branching	$\tilde{\mathbf{x}}(\text{LP}_k)$	$z(\text{LP}_k)$	LB	Outcome
1	–	0	–	$(\frac{7}{2}, \frac{5}{2})$	$\frac{31}{2}$	$-\infty$	branched on x_1
2	1	1	$x_1 \leq 3$	$(3, \frac{37}{14})$	$\frac{100}{7}$	14	pruned by bound (removed from L)
3	1	1	$x_1 \geq 4$	$(4, \frac{5}{4})$	$\frac{29}{2}$	$-\infty$	branched on x_2
4	3	2	$x_2 \leq 1$	$(\frac{41}{10}, 1)$	$\frac{143}{10}$	$-\infty$	branched on x_1
5	3	2	$x_2 \geq 2$	–	–	$-\infty$	pruned by infeasibility
6	4	3	$x_1 \leq 4$	$(4, 1)$	14	14	pruned by integrality

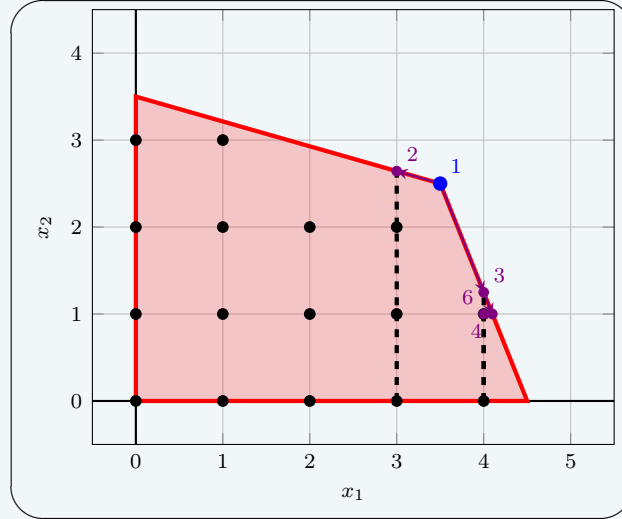


Figure 7.8: Execution A in the plane: the branching constraints (black, dashed) inside the regions of the parent nodes, and the optimal solutions of the solved nodes (violet), from LP_1 (blue).

Example 7.6: execution B: largest index and best bound

We execute again the B&B algorithm on the car example, now with:

1. the largest-index rule (lexicographic) as branching-variable selection strategy;
2. the best-bound rule as branching-node selection strategy (ties: smallest node index);
3. the down-branching node is solved first, then the up-branching node;
4. the second child is not solved when the skip rule applies;
5. the lower bound is initialized to $LB = -\infty$ (no heuristic algorithm is executed).

Each child is re-optimized with the dual simplex algorithm from the optimal dictionary of its parent, exactly as in [Example 7.5](#); here we report only the results.

- **Initialization**

As before, $\tilde{x}(LP_1) = (\frac{7}{2}, \frac{5}{2})$ with $z(LP_1) = \frac{31}{2}$, and LP_1 is inserted in the list L .

The list is $L = \{LP_1\}$, and the upper bound is the largest bound of the nodes in the list:

$$\underbrace{-\infty}_{LB} \leq z(ILP) \leq \underbrace{15}_{UB} \quad \left(UB = \max_{k \in L} UB(LP_k) = 15 \right).$$

- **Iteration 1**

The list is $L = \{LP_1\}$: the LP_1 problem is extracted and removed from L .

In $\tilde{\mathbf{x}}(\text{LP}_1)$ we have $\tilde{x}_1 = \frac{7}{2}$ and $\tilde{x}_2 = \frac{5}{2}$, both fractional; by the largest-index rule the branching variable is x_2 . The branching constraints are:

$$x_2 \leq \underbrace{\left\lfloor \frac{5}{2} \right\rfloor}_{=2} \quad \text{and} \quad x_2 \geq \underbrace{\left\lfloor \frac{5}{2} \right\rfloor + 1}_{=3}.$$

- Node 2 ($x_2 \leq 2$): $\tilde{\mathbf{x}}(\text{LP}_2) = (\frac{37}{10}, 2)$ with $z(\text{LP}_2) = \frac{151}{10}$. The solution is **fractional** and

$$\lfloor z(\text{LP}_2) \rfloor = \left\lfloor \frac{151}{10} \right\rfloor = 15 > \text{LB} = -\infty,$$

so the LP_2 problem is **inserted in the list L** .

- Node 3 ($x_2 \geq 3$): $\tilde{\mathbf{x}}(\text{LP}_3) = (\frac{7}{4}, 3)$ with $z(\text{LP}_3) = \frac{45}{4}$. The solution is **fractional** and

$$\lfloor z(\text{LP}_3) \rfloor = \left\lfloor \frac{45}{4} \right\rfloor = 11 > \text{LB} = -\infty,$$

so the LP_3 problem is **inserted in the list L** .

The list is $L = \{\text{LP}_2, \text{LP}_3\}$, and the upper bound is the largest bound of the nodes in the list:

$$\underbrace{-\infty}_{\text{LB}} \leq z(\text{ILP}) \leq \underbrace{15}_{\text{UB}} \quad \left(\text{UB} = \max_{k \in L} \text{UB}(\text{LP}_k) = 15 \right).$$

• Iteration 2

The list is $L = \{\text{LP}_2, \text{LP}_3\}$; following the best-bound rule, LP_2 is extracted and removed from L , since it has the largest optimal value:

$$z(\text{LP}_2) = \frac{151}{10}, \quad z(\text{LP}_3) = \frac{45}{4}.$$

In $\tilde{\mathbf{x}}(\text{LP}_2)$ the only integer variable with a fractional value is x_1 , which is the branching variable. The branching constraints are:

$$x_1 \leq \underbrace{\left\lfloor \frac{37}{10} \right\rfloor}_{=3} \quad \text{and} \quad x_1 \geq \underbrace{\left\lfloor \frac{37}{10} \right\rfloor + 1}_{=4}.$$

- Node 4 ($x_1 \leq 3$): $\tilde{\mathbf{x}}(\text{LP}_4) = (3, 2)$ with $z(\text{LP}_4) = 13$. The solution $\tilde{\mathbf{x}}(\text{LP}_4)$ is **integer**. Since $z(\text{LP}_4) = 13 > \text{LB} = -\infty$, the lower bound and the incumbent solution are updated:

$$\text{LB} \leftarrow 13, \quad \bar{\mathbf{x}} \leftarrow (3, 2).$$

Node 4 is pruned by integrality. Moreover, LP_3 is removed from the list L (pruned by bound), since now

$$\lfloor z(\text{LP}_3) \rfloor = \left\lfloor \frac{45}{4} \right\rfloor = 11 \leq \text{LB} = 13.$$

- Node 5 ($x_1 \geq 4$): $\tilde{\mathbf{x}}(\text{LP}_5) = (4, \frac{5}{4})$ with $z(\text{LP}_5) = \frac{29}{2}$. The solution is **fractional** and

$$\lfloor z(\text{LP}_5) \rfloor = \left\lfloor \frac{29}{2} \right\rfloor = 14 > \text{LB} = 13,$$

so the LP_5 problem is **inserted in the list** L .

The list is $L = \{\text{LP}_5\}$, and the upper bound is the largest bound of the nodes in the list:

$$\underbrace{13}_{\text{LB}} \leq z(\text{ILP}) \leq \underbrace{14}_{\text{UB}} \quad \left(\text{UB} = \max_{k \in L} \text{UB}(\text{LP}_k) = 14 \right).$$

• Iteration 3

The list is $L = \{\text{LP}_5\}$: the LP_5 problem is extracted and removed from L .

In $\tilde{\mathbf{x}}(\text{LP}_5)$ the only integer variable with a fractional value is x_2 , which is the branching variable. The branching constraints are:

$$x_2 \leq \underbrace{\left\lfloor \frac{5}{4} \right\rfloor}_{=1} \quad \text{and} \quad x_2 \geq \underbrace{\left\lfloor \frac{5}{4} \right\rfloor + 1}_{=2}.$$

- Node 6 ($x_2 \leq 1$): $\tilde{\mathbf{x}}(\text{LP}_6) = (\frac{41}{10}, 1)$ with $z(\text{LP}_6) = \frac{143}{10}$. The solution is **fractional** and

$$\lfloor z(\text{LP}_6) \rfloor = \left\lfloor \frac{143}{10} \right\rfloor = 14 > \text{LB} = 13,$$

so the LP_6 problem is **inserted in the list** L .

- Node 7 ($x_2 \geq 2$): the LP_7 problem is infeasible: pruned by infeasibility.

The list is $L = \{\text{LP}_6\}$, and the upper bound is the largest bound of the nodes in the list:

$$\underbrace{13}_{\text{LB}} \leq z(\text{ILP}) \leq \underbrace{14}_{\text{UB}} \quad \left(\text{UB} = \max_{k \in L} \text{UB}(\text{LP}_k) = 14 \right).$$

• Iteration 4

The list is $L = \{\text{LP}_6\}$: the LP_6 problem is extracted and removed from L .

In $\tilde{\mathbf{x}}(\text{LP}_6)$ the only integer variable with a fractional value is x_1 , which is the branching variable. The branching constraints are:

$$x_1 \leq \underbrace{\left\lfloor \frac{41}{10} \right\rfloor}_{=4} \quad \text{and} \quad x_1 \geq \underbrace{\left\lfloor \frac{41}{10} \right\rfloor + 1}_{=5}.$$

- Node 8 ($x_1 \leq 4$): $\tilde{\mathbf{x}}(\text{LP}_8) = (4, 1)$ with $z(\text{LP}_8) = 14$. The solution $\tilde{\mathbf{x}}(\text{LP}_8)$ is **integer**. Since $z(\text{LP}_8) = 14 > \text{LB} = 13$, the lower bound and the incumbent solution are updated:

$$\text{LB} \leftarrow 14, \quad \bar{\mathbf{x}} \leftarrow (4, 1).$$

Node 8 is pruned by integrality.

– **Skip rule.** After Node 8 the lower bound satisfies

$$\lfloor z(\text{LP}_6) \rfloor = \left\lfloor \frac{143}{10} \right\rfloor = 14 \leq \text{LB} = 14,$$

so no solution in the region of LP_6 can improve the incumbent solution: the up-branching child ($x_1 \geq 5$) is **not solved** and it does not receive a node index.

The list is empty, so the upper bound is equal to the lower bound:

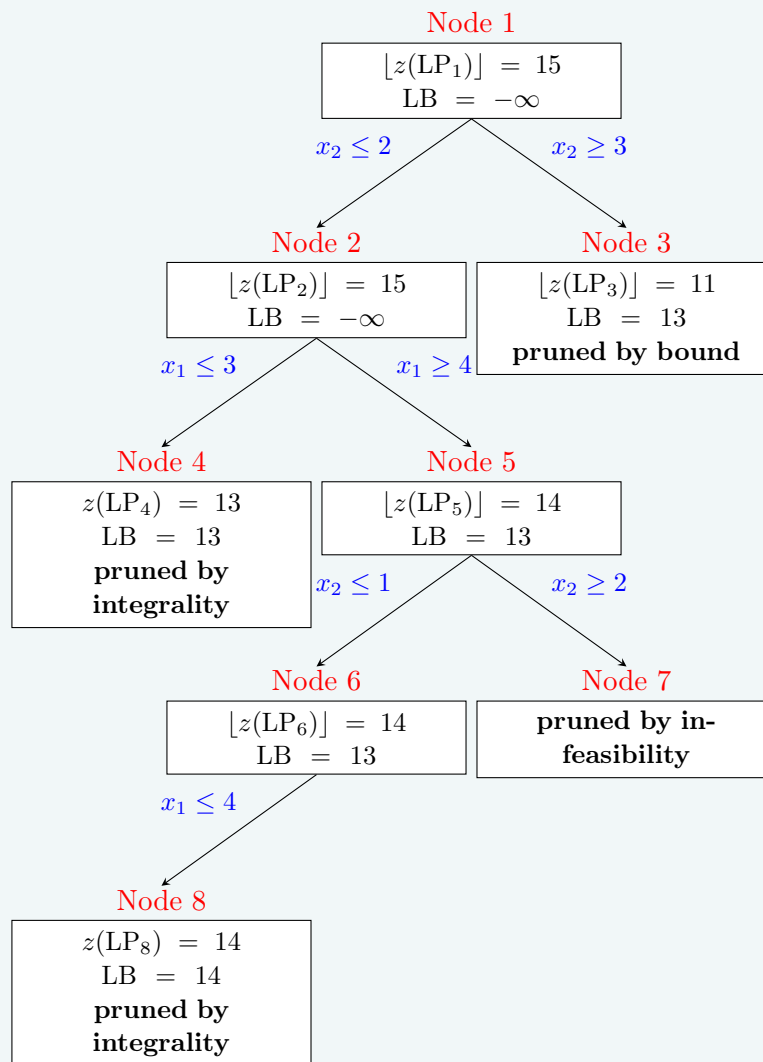
$$\underbrace{14}_{\text{LB}} \leq z(\text{ILP}) \leq \underbrace{14}_{\text{UB}} \quad (\text{UB} = \text{LB} = 14).$$

- **Termination**

The list L is empty: $\text{LB} = \text{UB} = 14$ and the incumbent solution is optimal:

$$\tilde{x}_1 = 4, \quad \tilde{x}_2 = 1, \quad z(\text{ILP}) = 14.$$

This is the final branch-and-bound tree:



Node	Parent	It.	Branching	$\tilde{\mathbf{x}}(\text{LP}_k)$	$z(\text{LP}_k)$	LB	Outcome
1	–	0	–	$(\frac{7}{2}, \frac{5}{2})$	$\frac{31}{2}$	$-\infty$	branched on x_2
2	1	1	$x_2 \leq 2$	$(\frac{37}{10}, 2)$	$\frac{151}{10}$	$-\infty$	branched on x_1
3	1	1	$x_2 \geq 3$	$(\frac{7}{4}, 3)$	$\frac{45}{4}$	13	pruned by bound (removed from L)
4	2	2	$x_1 \leq 3$	$(3, 2)$	13	13	pruned by integrality
5	2	2	$x_1 \geq 4$	$(4, \frac{5}{4})$	$\frac{29}{2}$	13	branched on x_2
6	5	3	$x_2 \leq 1$	$(\frac{41}{10}, 1)$	$\frac{143}{10}$	13	branched on x_1
7	5	3	$x_2 \geq 2$	–	–	13	pruned by infeasibility
8	6	4	$x_1 \leq 4$	$(4, 1)$	14	14	pruned by integrality

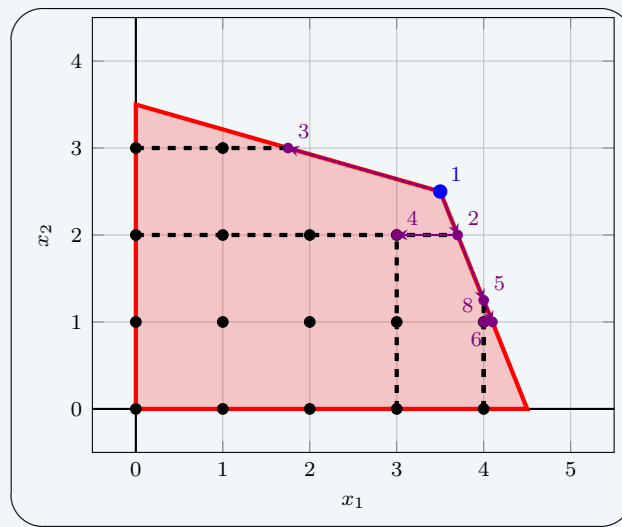


Figure 7.9: Execution B in the plane: branching constraints and optimal solutions of the solved nodes.

- Execution A solves 6 LP problems, execution B solves 8. Both certify the same optimal solution $\tilde{x}_1 = 4, \tilde{x}_2 = 1$ of value 14, which is unique.
- Execution B finds a first incumbent solution of value 13 earlier, but branching first on x_2 leads to a larger tree on this instance.

Example 7.7: heuristic initialization

We execute the B&B algorithm of [Example 7.5](#) (smallest index, LIFO), starting from a heuristic solution of the car example.

- The one-variable heuristic of Chapter 1 fixes x_1 as large as possible and $x_2 = 0$: it gives $\bar{\mathbf{x}} = (4, 0)$ of value 12. Starting execution A with $\text{LB} = 12$, the algorithm solves 6 LP problems, as many as without the heuristic: this lower bound is too weak to prune any node earlier.
- The heuristic that fixes first x_1 and then x_2 as large as possible gives $\bar{\mathbf{x}} = (4, 1)$ of

value 14. Starting with $LB = 14$, the root is not pruned, since $\lfloor z(LP_1) \rfloor = 15 > 14$, but both children are pruned by bound and the algorithm stops after 3 LP problems:

Node	Parent	It.	Branching	$\tilde{\mathbf{x}}(LP_k)$	$z(LP_k)$	LB	Outcome
1	–	0	–	$(\frac{7}{2}, \frac{5}{2})$	$\frac{31}{2}$	14	branched on x_1
2	1	1	$x_1 \leq 3$	$(3, \frac{37}{14})$	$\frac{100}{7}$	14	pruned by bound
3	1	1	$x_1 \geq 4$	$(4, \frac{5}{4})$	$\frac{29}{2}$	14	pruned by bound

The incumbent solution given by the heuristic is optimal, and the branch-and-bound algorithm **certifies** its optimality.

7.8 Typical mistakes

Warning

- **Rounding the LP solution.** Rounding an optimal solution of the LP relaxation gives at best a feasible solution, in general not an optimal one. In the car example, rounding down $\tilde{\mathbf{x}}(LP) = (\frac{7}{2}, \frac{5}{2})$ gives $(3, 2)$ of value 13, while $z(ILP) = 14$ at $(4, 1)$.
- **Branching on the wrong variable.** The branching variable must be an integer variable with a fractional value: a variable with an integer value does not cut off $\tilde{\mathbf{x}}(LP_k)$.
- **Taking the upper bound from the last node.** The upper bound is the largest bound among the nodes of the list L , not the bound of the last node solved.
- **Pruning by bound only with a strict inequality.** A node with $UB(LP_k) = LB$ is pruned: its region can contain solutions as good as the incumbent one, but not better.
- **Numbering the skipped children.** A child that is not solved does not receive a node index.
- **Solving every node from scratch.** The answer is correct, but much slower: each child starts from the optimal dictionary of its parent, which is dual feasible.

7.9 Extensions

- The LP problems of the nodes can also be solved with the **bounded-variable simplex algorithm**, which handles the branching constraints as bounds on the variables instead of new constraints.
- In a **mixed** problem, where only some variables are integer, one branches only on the integer variables and the bound of a node is $z(LP_k)$, without rounding.

7.10 In summary

In summary

- A B&B algorithm divides the feasible region with branching constraints on integer variables with fractional values, and it never loses a feasible solution.
- Every node is pruned by infeasibility, by integrality or by bound, or it waits in the list; the bound of a node is $\lfloor z(\text{LP}_k) \rfloor$ only if all the variables and costs are integer.
- The lower bound comes from the incumbent solution, the upper bound from the nodes of the list; when the list is empty they coincide, and the incumbent solution is certified optimal.
- Each child is re-optimized with the dual simplex algorithm from the dictionary of its parent.
- The selection strategies and a good heuristic solution change the size of the tree, not the optimal value.

Dashboard of the car example. The lower and upper bounds on $z(\text{ILP})$ found so far in this module:

Chapter	Lower bound	Upper bound	From
Primal and dual bounds	14	15	heuristic solution $(4, 1)$, and $\lfloor z(\text{LP}) \rfloor$ with $z(\text{LP}) = \frac{31}{2}$
Cutting plane algorithm	14	14	5 Gomory fractional cuts
This chapter, execution A	14	14	6 LP problems solved
This chapter, execution B	14	14	8 LP problems solved
This chapter, with LB = 14	14	14	3 LP problems solved

In the car example the algorithm closes the gap between LB = 13 (rounded LP solution) and UB = 15 (LP relaxation) by solving 6 LP problems.

Branch-and-cut

Prerequisites

- From this module, the chapters «Valid inequalities» and «Cover inequalities»: valid inequalities, and the separation of a violated inequality of a family.
- From this module, the chapter «Cutting plane algorithm»: Gomory fractional cuts, and the re-optimization of the LP problem with the dual simplex algorithm after the insertion of a cut.
- From this module, the chapter «LP-based branch-and-bound»: the list of LP problems, the pruning criteria, and the branching-variable and branching-node selection strategies.

Objectives

At the end of this chapter you will be able to:

- explain how cutting planes and branching operations complement each other;
- distinguish global and local valid inequalities, and say where a cut can be used;
- describe cut-and-branch and branch-and-cut algorithms, and the role of the cut pool;
- execute a cut-and-branch algorithm with Gomory cuts by hand on a small ILP problem and compare the size of the tree with and without cuts;
- generate a local cut at a branching node and re-optimize with the dual simplex algorithm.

8.1 Branch-and-cut algorithms

Definition 8.1: branch-and-cut algorithm

Given an ILP problem in canonical maximization form with integer input data, a **branch-and-cut algorithm** is a branch-and-bound algorithm in which the LP problems of the branching nodes are strengthened with cutting planes before branching.

- A **cutting plane algorithm** moves the LP relaxation towards the convex hull of the integer solutions, but Gomory fractional cuts lose effectiveness after the first iterations (**tailing off**).
- A **branch-and-bound algorithm** partitions the feasible region into smaller LP problems, but started from a weak LP relaxation it needs a large tree.

Cuts decrease the bounds of the nodes, so more nodes are pruned by bound; branching guarantees termination when the cuts are no longer effective.

- All modern MIP solvers are based on branch-and-cut algorithms (see the chapter «What a MIP solver does»).

8.2 Global and local valid inequalities

- In a branch-and-bound algorithm, the LP_k problem of a branching node k is the LP relaxation with the branching constraints of the path from the root to node k .
- We denote by F the set of integer feasible solutions of the ILP problem, and by $F_k \subseteq F$ the set of those that satisfy the branching constraints of node k . At the root node $F_1 = F$.

Definition 8.2: global and local valid inequalities

A linear inequality $\alpha'x \leq \beta$ is called:

1. a **globally valid inequality** (or **global cut**) if it is satisfied by all the solutions in F ;
2. a **locally valid inequality** for the branching node k (or **local cut**) if it is satisfied by all the solutions in F_k .

- Since $F_k \subseteq F$, every global cut is also a local cut of every node. The converse is false: a local cut can remove integer feasible solutions that belong to other branches of the tree.
- A local cut of node k is valid for all the nodes of the subtree rooted at k , and it must not be used outside this subtree.

A cut derived only from the original constraints (and from other global cuts) is **global**. A cut whose derivation uses a branching constraint is, in general, only **local**.

- A Gomory fractional cut generated at the root node is global: it is derived from the original constraints alone, and so is an inequality of a family (such as the cover inequalities of a constraint with positive coefficients) separated from them. A Gomory fractional cut generated by a row that contains the slack variable of a branching constraint with a non-zero coefficient is treated as local.

8.3 Cut-and-branch and branch-and-cut

Definition 8.3: cut-and-branch algorithm

A **cut-and-branch algorithm** executes a cutting plane algorithm at the root node for a limited number of **rounds of cuts**, adds the cuts found to the ILP problem, and then executes a branch-and-bound algorithm on the strengthened problem. No cuts are generated at the other nodes.

- All the root cuts are global: at the root node no branching constraint exists.
- In a **branch-and-cut algorithm** cuts are generated also at the other nodes (at all of them, or at some of them, for example every ten levels of the tree). The cuts can be global or local.
- At each node, the generation of cuts stops when the solution of the LP_k problem is integer or the node can be pruned; when no violated cut is found; when a maximum number of rounds is reached; or when the decrease of $z(LP_k)$ in the last rounds is small (tailing off).

8.3.1 The cut pool

- Cuts are stored in a data structure called the **cut pool**. Before calling the (more expensive) cut-generation procedures, the algorithm checks whether a cut of the pool is violated by $\tilde{x}(LP_k)$.
- Global cuts are stored in a **global pool**, usable at every node. Local cuts are stored with their node and are inherited only by its descendants.
- Each cut adds a constraint to the LP problems. Cuts that stay inactive for many iterations are removed from the LP problem and kept only in the pool.

8.3.2 High-level pseudocode

- The branch-and-cut algorithm has the pseudocode of the branch-and-bound algorithm (same branching operations, same pruning criteria, same list L): only the **Branching Node Solution Sub-Procedure** changes.

Algorithm

Input: An LP_k problem associated with a branching node, the cut pool, and a maximum number R of rounds of cuts.

Step 1: Bounding operations

Solve the LP_k problem.

- (a) if the LP_k problem is infeasible, then **STOP**.
- (b) if $\lfloor z(LP_k) \rfloor \leq LB$, then **STOP**.
- (c) if $\tilde{x}(LP_k)$ is integer, then update the **lower bound** and the **incumbent solution**, remove from the list L all LP problems with $\lfloor z(LP) \rfloor \leq LB$ and **STOP**.

Step 2: Cut generation

If R rounds of cuts have been executed at node k , go to Step 3. Otherwise:

- (a) look in the cut pool for cuts violated by $\tilde{x}(LP_k)$; if none is found, generate new violated cuts and store them in the pool (global cuts in the global pool, local cuts with node k);
- (b) if no violated cut exists, go to Step 3. Otherwise add the cuts to the LP_k problem, re-optimize it and go to Step 1.

Step 3: Insertion into the list

Insert the LP_k problem (with its cuts) into the list L and **STOP**.

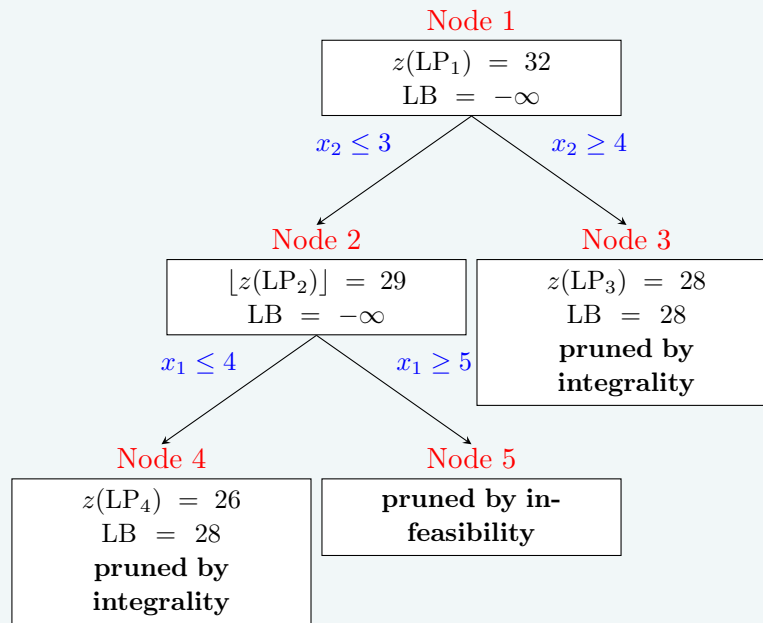
- With $R = 0$ at every node except the root we obtain a cut-and-branch algorithm; with $R = 0$ at every node we obtain the branch-and-bound algorithm.
- The pseudocode does not say how the cuts are generated or how the LP problem is re-optimized. After the insertion of a cut the dictionary is dual feasible, so the **dual simplex algorithm** starts from the optimal dictionary of the previous problem, as after a branching constraint.

8.4 Worked example: cut-and-branch with Gomory cuts

- We use the problem $P10$ of the catalogue of MIP problems: maximize $5x_1 + 2x_2$ subject to $-x_1 + 3x_2 \leq 12$, $x_1 + 2x_2 \leq 12$, $3x_1 - 2x_2 \leq 8$, $x_1, x_2 \in \mathbb{Z}_{\geq 0}$. Its unique optimal solution is $\tilde{x}(\text{ILP}) = (4, 4)$, with $z(\text{ILP}) = 28$.
- The cuts are the Gomory fractional cuts of the chapter on the cutting plane algorithm, with the same rule: the row with the largest fractional right-hand side, and the smallest index in case of a tie.
- Since the data are integer, the bound of a node is $\lfloor z(\text{LP}_k) \rfloor$. All the executions use the same rules: the smallest-index rule as branching-variable selection strategy, the LIFO rule as branching-node selection strategy, the down-branching node solved first, and $\text{LB} = -\infty$ at the start.

Example 8.1: branch-and-bound without cuts

The LP_1 problem is the LP relaxation of $P10$. An optimal solution is $\tilde{x}(\text{LP}_1) = (5, \frac{7}{2})$, with $z(\text{LP}_1) = 32$ and $\text{UB} = 32$. The branch-and-bound algorithm solves 5 LP problems:



Node	Parent	Branching	$\tilde{x}(\text{LP}_k)$	$z(\text{LP}_k)$	LB	Outcome
1	–	–	$(5, \frac{7}{2})$	32	$-\infty$	branching on x_2
2	1	$x_2 \leq 3$	$(\frac{14}{3}, 3)$	$\frac{88}{3}$	$-\infty$	branching on x_1
3	1	$x_2 \geq 4$	$(4, 4)$	28	28	pruned by integrality
4	2	$x_1 \leq 4$	$(4, 3)$	26	28	pruned by integrality
5	2	$x_1 \geq 5$	–	–	28	pruned by infeasibility

Example 8.2: cut-and-branch with one Gomory cut

- **Root: cut generation.** The two rows of the optimal dictionary of LP_1 with a fractional right-hand side, the rows of x_2 and of x_3 , have the same right-hand side $\frac{1}{2}$; we use the row of x_2 , whose index is smaller. With the slack variables $x_4 = 12 - x_1 - 2x_2$ and $x_5 = 8 - 3x_1 + 2x_2$ the cut is

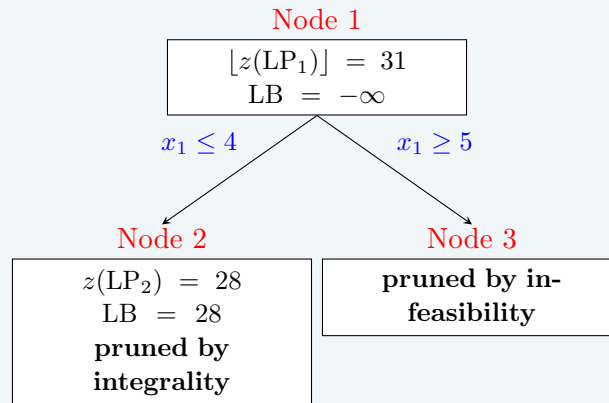
$$\frac{3}{8}x_4 + \frac{7}{8}x_5 \geq \frac{1}{2} \quad \Longleftrightarrow \quad 3x_1 - x_2 \leq 11.$$

It is a global cut, and it cuts off $\tilde{x}(LP_1)$: $3 \cdot 5 - \frac{7}{2} = \frac{23}{2} > 11$.

- **Root: re-optimization.** We add the cut to the LP_1 problem and we re-optimize it with the dual simplex algorithm. An optimal solution is

$$\tilde{x}(LP_1) = \left(\frac{34}{7}, \frac{25}{7}\right), \quad z(LP_1) = \frac{220}{7}, \quad \text{UB} = 31.$$

- **Branch-and-bound.** The branch-and-bound algorithm now starts from this LP_1 problem. Every node inherits the cut. It solves 3 LP problems instead of 5:



Node	Parent	Branching	$\tilde{x}(LP_k)$	$z(LP_k)$	LB	Outcome
1	–	–	$(\frac{34}{7}, \frac{25}{7})$	$\frac{220}{7}$	$-\infty$	branching on x_1
2	1	$x_1 \leq 4$	$(4, 4)$	28	28	pruned by integrality
3	1	$x_1 \geq 5$	–	–	28	pruned by infeasibility

- One Gomory cut reduces the number of LP problems from 5 to 3. The cut does not find the optimal solution: it moves the optimal solution of the root LP problem, so that the algorithm branches on x_1 and the node $x_1 \leq 4$ gives the optimal solution at once.
- A second round of cuts, in the same cut-and-branch algorithm, gives an integer solution at the root. The row of x_1 has the largest right-hand side $\frac{6}{7}$; with the slack variable $x_6 = 11 - 3x_1 + x_2$ of the first cut the new cut is

$$\frac{1}{7}x_4 + \frac{2}{7}x_6 \geq \frac{6}{7} \quad \Longleftrightarrow \quad x_1 \leq 4.$$

- The optimal solution of the new LP_1 problem is $(4, 4)$, with $z(LP_1) = 28$. The node is

pruned by integrality, and the algorithm stops with $LB = UB = 28$ after solving one LP problem.

Cuts added at the root	$z(LP_1)$	UB at the root	LP problems solved
none (branch-and-bound)	32	32	5
$3x_1 - x_2 \leq 11$	$\frac{220}{7}$	31	3
$3x_1 - x_2 \leq 11$ and $x_1 \leq 4$	28	28	1

- The stronger the cuts, the smaller the tree, until it reduces to the root node.
- This is not the rule. The table below reports the same algorithm on the problem $P2$ of the catalogue (maximize $x_1 + 3x_2$ subject to $-4x_1 + 10x_2 \leq 35$, $3x_1 + x_2 \leq 12$, $x_1, x_2 \in \mathbb{Z}_{\geq 0}$, with $z(ILP) = 14$) for an increasing number of rounds of Gomory cuts.

Rounds	Last cut added	$z(LP_1)$	UB at the root	LP problems solved
0	–	16	16	4
1	$-3x_1 + 10x_2 \leq 37$	$\frac{524}{33}$	15	4
2	$x_1 + 3x_2 \leq 15$	15	15	4
3	$-x_1 + 5x_2 \leq 19$	15	15	4
4	$x_2 \leq 4$	$\frac{44}{3}$	14	2
5	$x_1 + x_2 \leq 6$	14	14	1

- The first three cuts decrease $z(LP_1)$ from 16 to 15 but the tree has always 4 nodes; 5 rounds are needed to reach the root node. The upper bound improves slowly, as in the **tailing off** of the cutting plane algorithm.
- A cut-and-branch algorithm must decide when to stop cutting and start branching: the cost of each round is paid at every node of the tree.

8.5 Local cuts: an example

- We use the car example (x_1 sports cars and x_2 family cars): maximize $3x_1 + 2x_2$ subject to $10x_1 + 4x_2 \leq 45$, $4x_1 + 14x_2 \leq 49$, $x_1, x_2 \in \mathbb{Z}_{\geq 0}$. In the branch-and-bound algorithm without cuts, node 3 is the up-branching child of the root, with the branching constraint $x_1 \geq 4$ and the slack variable $x_5 = x_1 - 4$.
- The optimal dictionary of the LP_3 problem, with $\tilde{x}(LP_3) = (4, \frac{5}{4})$ and $z(LP_3) = \frac{29}{2}$, is:

$$\left\{ \begin{array}{lclcl} x_1 & = & 4 & + & x_5 \\ x_2 & = & \frac{5}{4} - \frac{1}{4}x_3 - \frac{5}{2}x_5 \\ x_4 & = & \frac{31}{2} + \frac{7}{2}x_3 + 31x_5 \\ z & = & \frac{29}{2} - \frac{1}{2}x_3 - 2x_5 \end{array} \right.$$

Example 8.3: a local cut at node 3

- **A simple local cut.** From $10x_1 + 4x_2 \leq 45$ and the branching constraint $x_1 \geq 4$ we have

$$4x_2 \leq 45 - 10x_1 \leq 45 - 10 \cdot 4 = 5 \implies x_2 \leq \frac{5}{4} \implies x_2 \leq 1.$$

- The inequality $x_2 \leq 1$ is valid for all the solutions in F_3 , but it is **not globally valid**: the integer feasible solution $(3, 2)$, of value 13, violates it. It is a local cut of node 3.
- We add it to the LP_3 problem, with the slack variable $x_6 = 1 - x_2$, and we re-optimize.
- The cut is inserted with the slack variable $x_6 \geq 0$. The new row is

$$x_6 = -\frac{1}{4} + \frac{1}{4}x_3 + \frac{5}{2}x_5.$$

- The dictionary below is **dual feasible** but not primal feasible, since $\bar{b}_6 = -\frac{1}{4} < 0$:

$$\left\{ \begin{array}{rclcl} x_1 & = & 4 & + & x_5 \\ x_2 & = & \frac{5}{4} & - & \frac{1}{4}x_3 & - & \frac{5}{2}x_5 \\ x_4 & = & \frac{31}{2} & + & \frac{7}{2}x_3 & + & 31x_5 \\ x_6 & = & -\frac{1}{4} & + & \frac{1}{4}x_3 & + & \frac{5}{2}x_5 \\ z & = & \frac{29}{2} & - & \frac{1}{2}x_3 & - & 2x_5 \end{array} \right.$$

- **Dual pivoting operation.** The pivot row is x_6 , the only basic variable with a negative value. The ratios of the non-basic variables with $\bar{a}_{6j} < 0$ are

$$\frac{\bar{c}_3}{\bar{a}_{6,3}} = \frac{-\frac{1}{2}}{-\frac{1}{4}} = 2, \quad \frac{\bar{c}_5}{\bar{a}_{6,5}} = \frac{-2}{-\frac{5}{2}} = \frac{4}{5}.$$

- The pivot column is x_5 , with $s_m = \frac{4}{5}$, and the objective value becomes $\bar{z} + s_m \bar{b}_6 = \frac{29}{2} + \frac{4}{5} \cdot \left(-\frac{1}{4}\right) = \frac{143}{10}$. The new dictionary is primal and dual feasible, hence optimal:

$$\left\{ \begin{array}{rclcl} x_1 & = & \frac{41}{10} & - & \frac{1}{10}x_3 & + & \frac{2}{5}x_6 \\ x_2 & = & 1 & & & - & x_6 \\ x_4 & = & \frac{93}{5} & + & \frac{2}{5}x_3 & + & \frac{62}{5}x_6 \\ x_5 & = & \frac{1}{10} & - & \frac{1}{10}x_3 & + & \frac{2}{5}x_6 \\ z & = & \frac{143}{10} & - & \frac{3}{10}x_3 & - & \frac{4}{5}x_6 \end{array} \right.$$

- The solution $\hat{x}(LP_3) = (\frac{41}{10}, 1)$, with value $\frac{143}{10}$, is the same that the down-branching child of node 3 gives: here the local cut works as a branching constraint, without creating a new node.

Example 8.4: a Gomory fractional cut at node 3

- The row of x_2 of the dictionary of node 3 gives the Gomory fractional cut

$$\frac{1}{4}x_3 + \frac{1}{2}x_5 \geq \frac{1}{4}$$

- It contains the slack variable x_5 of the branching constraint $x_1 \geq 4$, so its derivation uses this constraint and the algorithm treats it as a **local cut**.
- In the space of the original variables ($x_3 = 45 - 10x_1 - 4x_2$, $x_5 = x_1 - 4$, multiplied by 4) it reads $2x_1 + x_2 \leq 9$. Enumerating the 16 integer feasible solutions we see that this inequality is valid for all of them.
- The algorithm cannot make this check at a node, so it keeps the cut local.

8.6 Practical aspects

8.6.1 Cut selection

- At each round many violated cuts can be available; adding all of them makes the LP problems large and slow to re-optimize. A branch-and-cut algorithm selects a subset.
- The **efficacy** of a cut $\alpha'x \leq \beta$ at the fractional solution $\tilde{x}(\text{LP}_k)$ is the Euclidean distance of $\tilde{x}(\text{LP}_k)$ from the hyperplane $\alpha'x = \beta$:

$$\text{efficacy} = \frac{\alpha' \tilde{x}(\text{LP}_k) - \beta}{\|\alpha\|_2}.$$

- Unlike the violation $\alpha' \tilde{x}(\text{LP}_k) - \beta$, the efficacy does not change if the cut is multiplied by a positive constant. Selection rules combine efficacy with a measure of the **parallelism** of the cuts.
- For example, at $\tilde{x}(\text{LP}_1) = (5, \frac{7}{2})$ of $P10$ the cut $3x_1 - x_2 \leq 11$ has violation $\frac{1}{2}$ and efficacy $\frac{1}{2\sqrt{10}} \approx 0.158$, while the same cut multiplied by 2, $6x_1 - 2x_2 \leq 22$, has violation 1 and the same efficacy $\frac{1}{\sqrt{40}} \approx 0.158$.

8.6.2 Which cuts

- Any family of valid inequalities with a separation procedure can be used: Gomory fractional cuts (for ILP problems with integer data), cover inequalities for the constraints $\sum_j a_j x_j \leq b$ with positive coefficients of a MIP problem with binary variables, and families that exploit the structure of a given model.
- Gomory fractional cuts come from the rows of the optimal dictionary, so they are always available. In floating-point arithmetic their coefficients are affected by rounding errors, and solvers discard the cuts with very large or very small coefficients.

8.6.3 Upper bound and gap

- As in the branch-and-bound algorithm, if the list L is not empty we have

$$z(\text{ILP}) \leq \text{UB} = \max_{k \in L} \{ \text{UB}(\text{LP}_k) \},$$

where $z(\text{LP}_k)$ is now the optimal value of the LP_k problem after the cuts.

- Cuts decrease the values $z(\text{LP}_k)$ and, accordingly, the upper bound UB and the optimality gap $\frac{\text{UB}-\text{LB}}{|\text{LB}|}$.

8.7 Typical mistakes

Warning

- **Using a local cut outside its subtree.** A cut derived with a branching constraint of node k is valid only in the subtree of node k : in another branch it can remove an optimal solution.
- **Treating a cut as global without a proof.** A Gomory fractional cut generated at a node can be global by chance (Example 8.4); the algorithm keeps it local unless its derivation does not use branching constraints.
- **Adding every violated cut.** Too many cuts make the LP problems slow; a few effective cuts per round are better, and inactive cuts are removed.
- **Believing that every cut reduces the tree.** On $P2$ the first three Gomory cuts leave the tree at 4 nodes (table of Section 8.4); the quality of the cuts matters.
- **Choosing the cut with the largest efficacy as the strongest.** Efficacy is measured at the current point only: a dominated cut can have a larger efficacy at that point.
- **Forgetting that cuts decrease the bound, not the optimal value.** The optimal value $z(\text{ILP})$ never changes: only $z(\text{LP}_k)$ and the upper bound do.

8.8 In summary

In summary

- A branch-and-cut algorithm strengthens the LP problem of every node with cuts before branching.
- A global cut is valid for all the integer solutions; a local cut only for those of a subtree, and it is stored with its node.
- A cut-and-branch algorithm generates cuts only at the root; the cuts are then global.
- After a cut, the LP problem is re-optimized with the dual simplex algorithm from the previous dictionary.
- Cuts reduce the size of the tree, but only effective cuts do: on P_{10} , one Gomory cut reduces the tree from 5 to 3 LP problems, and two cuts to 1.

Dashboard of P_{10} . The lower and upper bounds on $z(\text{ILP}) = 28$ found in this chapter:

Stage	Lower bound	Upper bound	From
LP relaxation of P_{10}	—	32	$z(\text{LP}) = 32$, $\tilde{\mathbf{x}}(\text{LP}) = (5, \frac{7}{2})$
Branch-and-bound	28	28	5 LP problems solved
After the first Gomory cut	—	31	$z(\text{LP}_1) = \frac{220}{7}$
Cut-and-branch with one cut	28	28	3 LP problems solved
After the second Gomory cut	28	28	$\tilde{\mathbf{x}}(\text{LP}_1) = (4, 4)$ is integer, 1 LP problem solved

How MIP solvers combine these ideas is the topic of the chapter «MIP solvers».