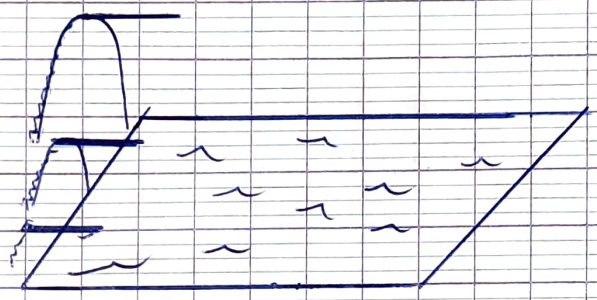
DIVING CONTEST
(BRANCH AND BOUND ALGORITHM)



We consider a diving competition with the following rules:

① There are 3 diving boards : 1m, 3m, 10m

② Jumping once off the 1m diving board earns 1 point
" " 3m " " 2 points
" " 10m " " 3 points

③ The aim is to win as many points as possible

④ Time constraint : competition lasts 10 min (= 600 s)
- Climb ladder, jump, swim takes 35 s for 1m board
- " " " 40 s " 3m "
- " " " 65 s " 10m "

⑤ • Lifeguard forbids more 10m dives than 1m dive
• " 3m " 1m dive

Problem : find the best number of dives (1m, 3m, 10m) to maximize the number of points.

Equivalently, solve:

$$\begin{cases} \text{Max } x + 2y + 3z \\ \text{s.t. } 35x + 40y + 65z \leq 600 \\ z \leq x \\ y \leq x \\ x \geq 0 \quad y \geq 0 \quad z \geq 0 \end{cases}$$

$x, y, z \in \mathbb{Z}$

First, we solve the problem with $x, y, z \in \mathbb{R}$

We use the **SIMPLEX** method

(see Simplex - Algorithm - Linear - Programming)

STANDARD FORM :

$$\begin{cases} \text{Max} & x + 2y + 3z \\ \text{s.t.} & 35x + 40y + 65z + a = 600 \\ & -x + z + b = 0 \\ & -x + y + c = 0 \\ & x, y, z, \underbrace{a, b, c}_{\text{deviation variables}} \geq 0 \end{cases}$$

Let's choose the base (x, y, z)

It's a **base** because the matrix is

$$\begin{pmatrix} 35 & 40 & 65 \\ -1 & 0 & 1 \\ -1 & 1 & 0 \end{pmatrix}$$

and this matrix is invertible

And this base is **feasible** because

$$\begin{pmatrix} 35 & 40 & 65 \\ -1 & 0 & 1 \\ -1 & 1 & 0 \end{pmatrix}^{-1} \begin{pmatrix} 600 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 30/7 \\ 30/7 \\ 30/7 \end{pmatrix} \in (\mathbb{R}_+)^3$$

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 35 & 40 & 65 \\ -1 & 0 & 1 \\ -1 & 1 & 0 \end{pmatrix}^{-1} \begin{pmatrix} 600-a \\ -b \\ -c \end{pmatrix}$$

$$= \begin{pmatrix} \frac{1}{140} & -\frac{13}{28} & -\frac{2}{7} \\ \frac{1}{140} & -\frac{13}{28} & \frac{5}{7} \\ \frac{1}{140} & \frac{15}{28} & -\frac{2}{7} \end{pmatrix} \begin{pmatrix} 600-a \\ -b \\ -c \end{pmatrix}$$

$$= \begin{pmatrix} \frac{1}{140}(600-a) + \frac{13}{28}b + \frac{2}{7}c \\ \frac{1}{140}(600-a) + \frac{13}{28}b - \frac{5}{7}c \\ \frac{1}{140}(600-a) - \frac{15}{28}b + \frac{2}{7}c \end{pmatrix}$$

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} \frac{30}{7} - \frac{a}{140} + \frac{13}{28}b + \frac{2}{7}c \\ \frac{30}{7} - \frac{a}{140} + \frac{13}{28}b - \frac{5}{7}c \\ \frac{30}{7} - \frac{a}{140} - \frac{15}{28}b + \frac{2}{7}c \end{pmatrix}$$

$$x + 2y + 3z = \frac{180}{7} - \frac{9}{140}a - \frac{3}{14}b - \frac{2}{7}c$$

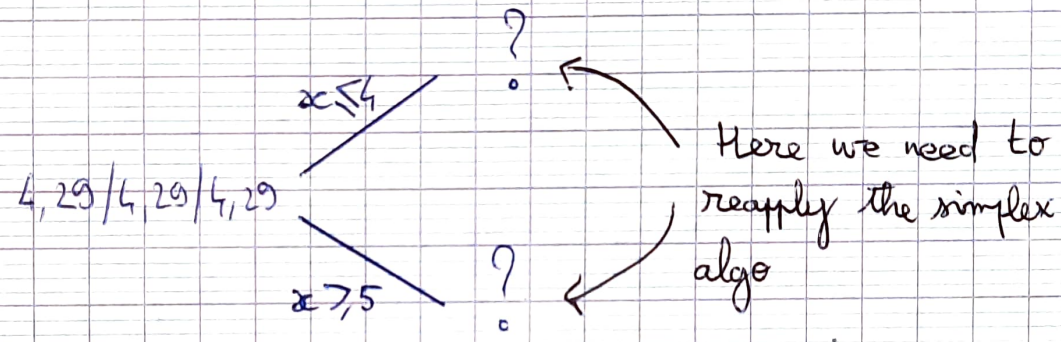
All reduced costs are negative, so the solution $(x, y, z) = \left(\frac{30}{7}, \frac{30}{7}, \frac{30}{7}\right)$ is optimal

$\frac{30}{7} \approx 4{,}29$

But this solutions is not made up of integers!

So now we are going to use Branch-and-Bound Algorithm.

First, we have to choose a variable (x, y or z) for branching. In practice, we always take the variable whose decimal part is closest to 0,5.
Here, x, y and z are equal at the beginning!
We can therefore <u>choose</u> x for branching



To go faster, let's use Julia and Gurobi.
I give you the code corresponding to this sub-problem:

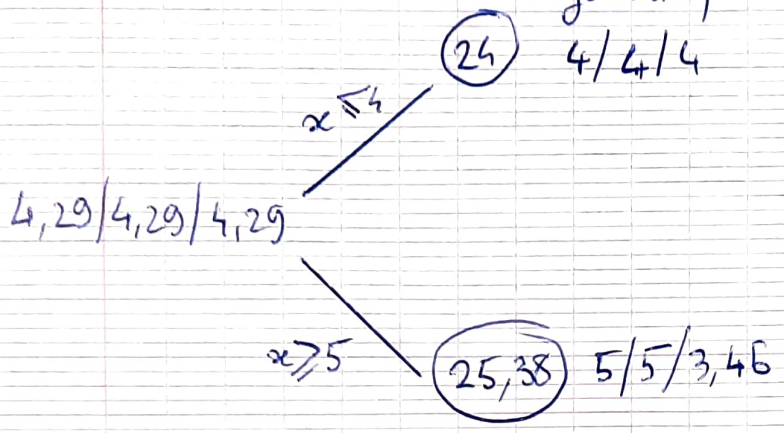
$$\begin{cases} \text{Max} \quad x + 2y + 3z \\ \text{st} \quad 35x + 40y + 65z \leq 600 \\ \qquad z \leq x \\ \qquad y \leq x \\ \qquad x \leq 4 \end{cases}$$

← additional constraint

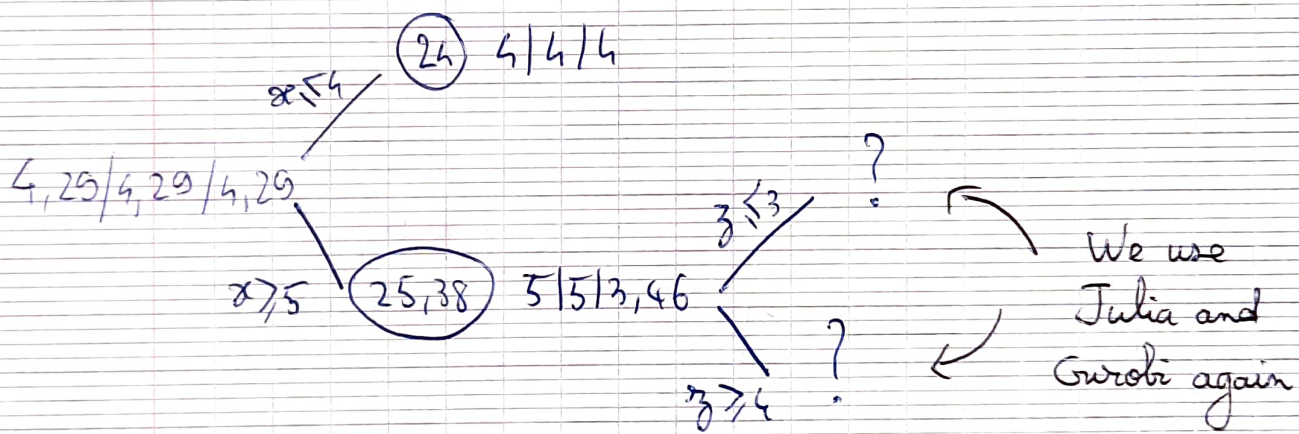
in the file <u>simplex.jl</u>

We obtain

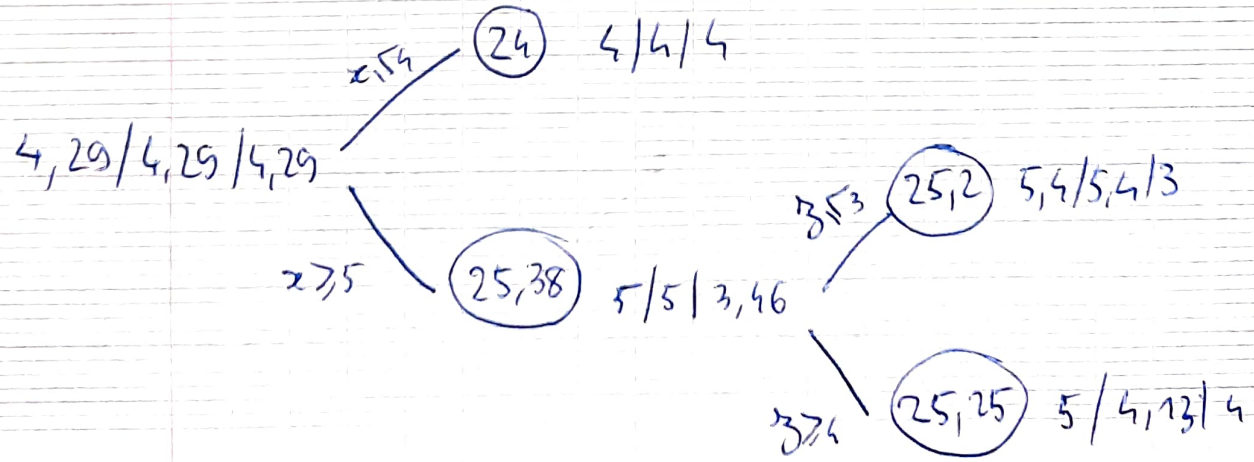
optimal value obtained with Gurobi (Simplex Algorithm)



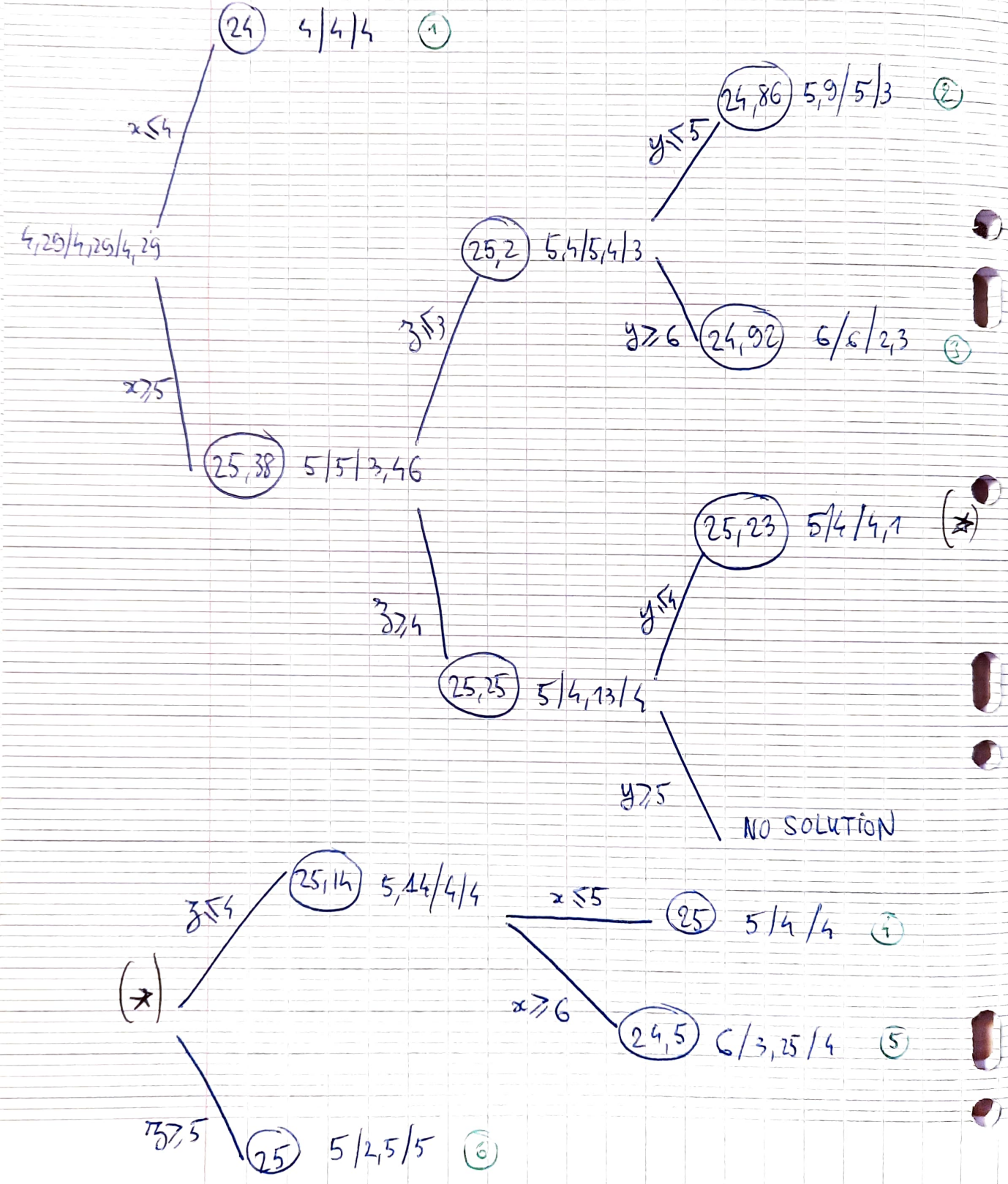
We do branching again with $z \leq 3$ and $z \geq 4$



We obtain:



And we continue (steps not detailed) to complete the tree



Note: we branch each time a node's solution is not integer. **BUT** we also let our intuition guide us and we don't develop every branch.

How to conclude?

In (4) we have an integer solution with value 25 $\Rightarrow$ (1); (2); (3); (5); (6) become useless because they have lower optimum values. Here we used **BOUND** property to eliminate these nodes.

This is why we call this algorithm BRANCH AND BOUND.

Note: the important fact is that when you are in a given node, the child nodes will necessarily have a **SMALLER** optimal value because they have **MORE** constraints.

The optimal solution is five 1m, four 3m and four 10m with a total of points of 25 points!

Note: of course you can also use Gurobi directly to solve the global problem, just specify that variables are "Int"

```
@variable(model, x>=0, Int)
```