Loops with constant objective function

With the simplex method, we move from a ~~base~~ B to B^{-1}, decreasing the objective function

All basis B have different cost ~~and~~ are different.

\# of possible bases $\approx \binom{n}{m} \rightarrow$ Large but <u>finite</u> number

For some iterations, the change in the obj function can be 0

It could happen that we revisit the same basis (the objective function is staying the same)

$\Downarrow$

Creating a loop (infinite)

$\downarrow$

Can happen!!! (very often)

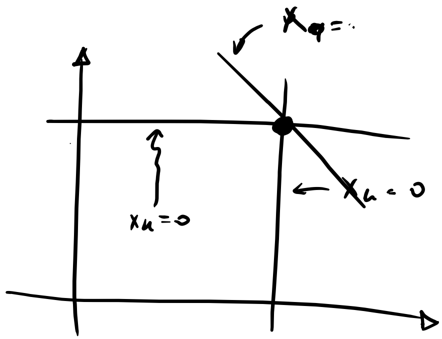
$$|\Delta z| = |\bar{c}_h| \vartheta \rightarrow \min \left\{ \frac{\bar{b}_i}{\bar{a}_{ih}} : a_{ih} > 0 \right\}$$

$|\bar{c}_h| > 0$

$$|\Delta z| = 0 \iff \bar{b}_t = 0, \ t = \operatorname{argmin} \left\{ \frac{\bar{b}_i}{\bar{a}_{ih}} : \bar{a}_{ih} > 0 \right\}$$

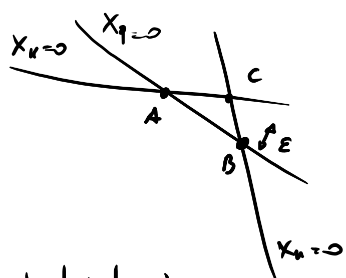
if $\bar{b}_t = 0$, the cost won't change

Degeneracy



finding ~~oth~~ vertex between $x_h = 0$ and $x_u = 0$ $\Rightarrow$ could be that $x_q = 0$ too in that point

I can move the line of x_q we can see better what happens:



$\rightarrow$ if $\varepsilon = 0 \rightarrow$ I can move from B to C and possibly back to B

$\downarrow$

loop

I could remove the redundant constraint and have the same polyhedron

We can't remove constraints that have $x_i = 0$ $\Rightarrow$ We may change the polyhedron in more dimensions.

How to deal with degeneracy:

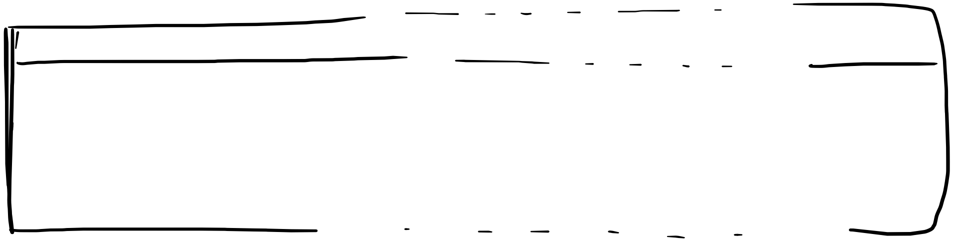
1) Random pivot $\rightarrow$ $P(\text{loop}) \sim 0$

2) Move randomly the costraints when we find a degeneracy

at some point we must go back $\Rightarrow$ too early = still in the loop

3) Bland's rule $\rightarrow$ Guarantees no loop

Revised simplex method



HUGE TABLEAU

Probable constraints : $x_i + x_j \le 1$ $\Rightarrow$ Lots of 0s

I don't keep the matrix in memory but a sparse version of the tableau

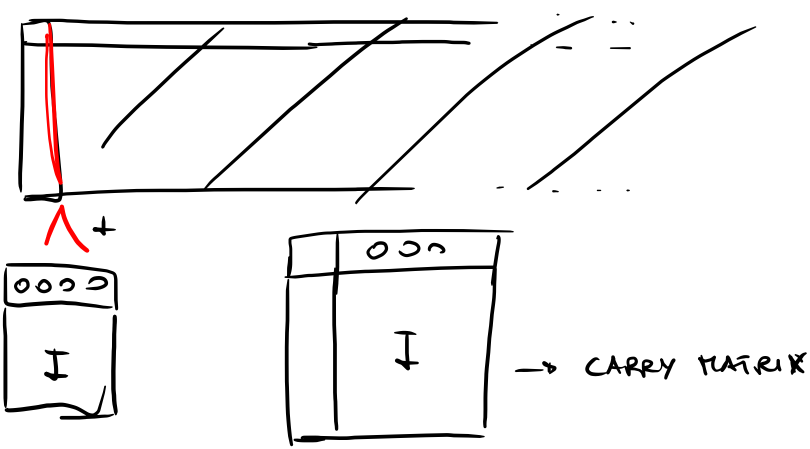
position and value of the non-zero elements

Usually we store columns : the cost and elements $\neq 0$

$$A_j = \begin{bmatrix} c_j \\ 0 \\ \vdots \\ 1 \\ 0 \\ 5 \\ \vdots \\ 0 \end{bmatrix} \begin{array}{c} \\ \\ \\ k \\ \\ t \\ \\ \\ \end{array} \longrightarrow \begin{bmatrix} c_j \\ (k : 1) \\ (t : 5) \end{bmatrix} \begin{array}{l} \\ \to (\text{position } k) = 1 \\ \to (\text{position } t) = 5 \end{array}$$

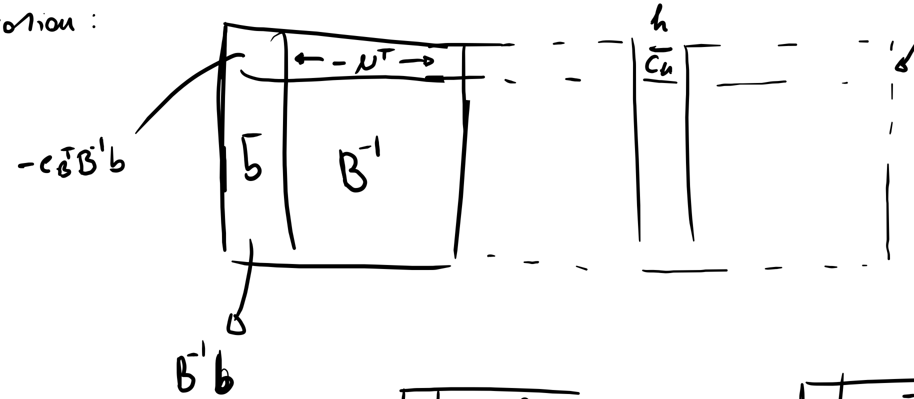
After lots of iterations → lots of non-zero... ⇒ can't keep the sparse format

I cut the tableau and store only the left part



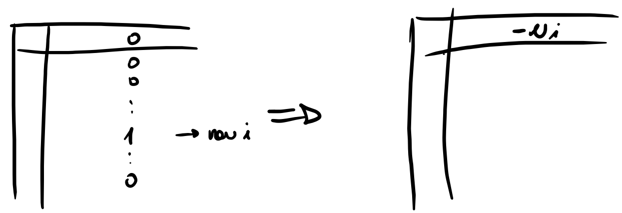
the rest of the tableau can be computed from the carry matrix

First iteration:



$z^T = c^T - \nu^T A$

$\nu^T = c_B^T B^{-1}$

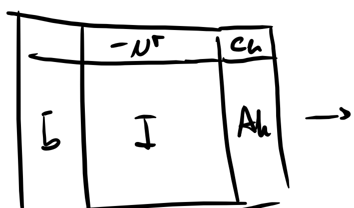


stored at the beginning

$\bar{c}_i = c_i - \nu^T A_i$

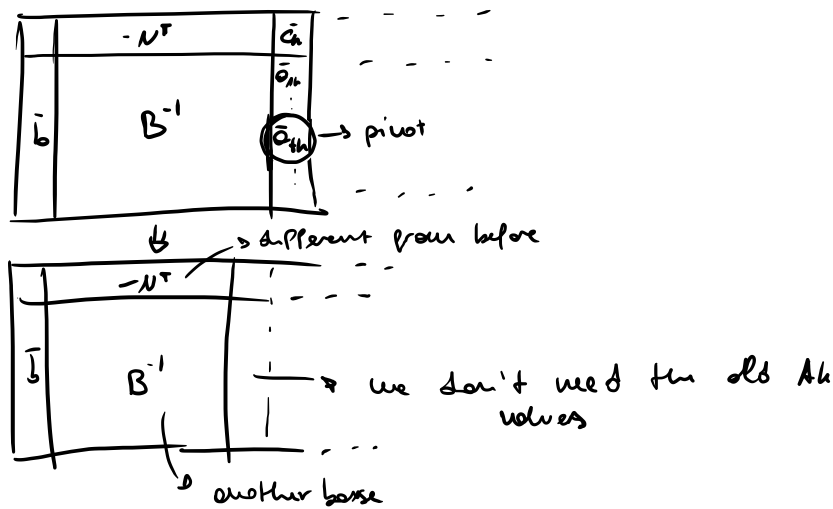
product is fast if the matrix is sparse

I will find $\bar{c}_h < 0 \Rightarrow$ Compute A_h



from here I can search and make the pivot operation

I will need to update just the $m \times m$ matrix



Tipically : $\underbrace{\#\text{ pivot operation}}_{(\#\text{ iterations})}$ needed $\sim 3m$ $\rightarrow$ $\ll n$

Most of the variables will probably remain non-basis