

1) On cherche  $A, b, c, x$  tels que le problème

s'énonce maximiser  $c^T x$

où  $Ax \leq b$ .

On peut prendre  $c = (r_{n,k,m})_{\substack{k \leq K \\ n \leq N \\ m \leq M}}$  et  $x = (x_{n,k,m})$

De sorte à ce que  $c^T x = \sum_{n,k,m} r_{n,k,m} x_{n,k,m}$

Sachant  $\sum_{n,k,m} p_{n,k,m} x_{n,k,m} \leq p$  (1)

et  $\sum_{k,m} x_{n,k,m} \leq 1 \quad \forall n \leq N$  (2)

Donc  $A^{(1)T} = (p_{n,k,m})_{\substack{n \leq N \\ k \leq K \\ m \leq M}}$   $b^{(1)} = p$

$A^{(2)} = (1)_{\substack{n \leq N \\ k \leq K \\ m \leq M}}$   $b^{(2)} = (1)_{n \leq N}$

On peut voir les matrices en 2D de la façon suivante:

$$A = \left( \frac{A^1}{A^2} \right)^{\overleftarrow{\overrightarrow{1}}} \quad b = \begin{pmatrix} p \\ 1 \\ \vdots \\ 1 \end{pmatrix} \quad x = \begin{pmatrix} \square \\ \square \\ \vdots \\ \square \end{pmatrix}^{\overleftarrow{\overrightarrow{f_N}}} \quad \text{NMK}$$

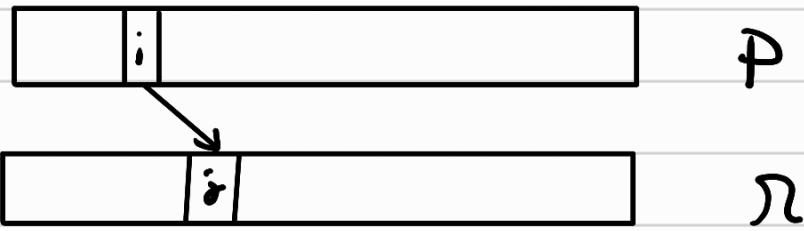
Avec  $A' = \begin{pmatrix} 1 & 1 & \dots & 1 \end{pmatrix}$

$$A^2 = \begin{pmatrix} 1 & 1 & \dots & 1 \\ 1 & 1 & \dots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \dots & 1 \end{pmatrix}$$

$\xrightarrow{\text{N blocs}}$

- 2) On vérifie simplement que  $\forall n, k, m, p_{n,k,m} \leq p$ .  
et s'il en reste à la fin.
- 3) For  $n$  in range ( $N$ ):  $\xrightarrow{\Theta(N)}$   $\xrightarrow{\Omega(KM \ln(KM))}$   
 trier les  $(p_{n,k,m})_{\substack{k \leq K \\ m \leq M}}$  et les  $(r_{n,k,m})_{\substack{k \leq K \\ m \leq M}}$   
 dans l'ordre croissant

Puis par courir les deux listes obtenues simultanément :



On élimine tous les couples  $(k, m) = j$  tels que

$$\pi[j] \leq \pi[i] = \pi_{n,k,m}$$

Puis on incrémente  $i$  en vérifiant que  $p[i]$  n'a pas été éliminé

$\Theta(KM)$  car c'est étant strictement croissant

$$\forall n, k, m = i$$

- $\rightarrow p_i$  a été éliminé si  $\exists n', k', m' = i'$
- $p_{i'} \leq p_i$  et  $n_{i'} > n_i$
- $\rightarrow$  Sinon  $p_i$  reste.

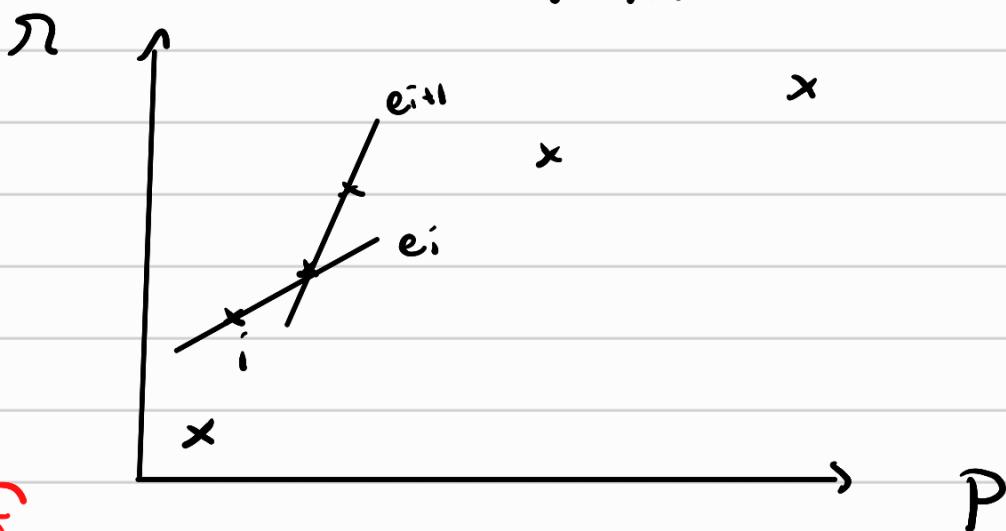
4) De même,

$\Theta(N)$

For  $n$  in range  $N$ :

On trie les  $p_{n, k, m}$

$\Theta(KM)$



les fonctions de  $p$  sont croissantes d'après 2)

On veut maintenant rendre cette fonction concave :

Tant que  $p_i$  n'est pas déjà éliminé ou  $i > KM$ ,

si  $e_i < e_{i+1}$  :

enlever  $p_{i+1}$

Sinon :

$i+1$

$\Theta(KM)$  car à chaque étape soit on traite un  $p_i$  en  $\Theta(n)$

5) c-f code

Greedy algorithm:

6) def greedy(instance):  $\Theta(NKM \ln(KN))$

→ On preprocess l'instance en utilisant les algorithmes des questions 2) 3) 4).

For  $n$  in range  $N$ :  $\Theta(N)$

$\Theta(M^2 \ln(M))$  | brier les  $p_{n,k,m}$  restants par ordre croissant

En faire  $n$  files de priorité qui correspondent au tri décroissant des  $e_n$  par chaque  $n$

$p\_restant = p$

$L = [(0, 0)]^* N \rightarrow$  Correspond à la dernière allocation pour chaque  $n$ .

While  $p\_restant > 0$  ou reste des possibilités d'allocation:

$(n, k, m) = \max_n (\text{tête } E[n]) \rightarrow$  on élimine au passage la tête correspondante.

$(k_-, m_-) = L[n]$

$$L[n] = (k, m)$$

$$P_{\text{restant}} := \left( P_{n,k,m} - P_{n,k_-,m_-} \right) \quad \begin{matrix} \downarrow \\ \text{Vault } 0 \text{ si } m_- = 0. \end{matrix}$$

$$X[n][k][m] = 1$$

if  $(k_-, m_-) \neq 0$ :

$$X[n][k_-][m_-] = 0$$

return  $X$ .

7) cf. code (LP solver utilisé: celui de scipy)

Algorithms for solving  
the ILP.

8) We will be using a 2D array for  
the DP, taking the parameters  $p$

and  $n$ .

$$DP = [0]^n \times N^p \times p_{\max}$$

for  $n$  in range  $N$ ,

for  $p$  in range  $p_{\max}$ :

$\Theta(N_p)$

$$DP(n, p) = \max_{k, m} \left( DP(n-1, p - p_{n, k, m}) + r_{n, k, m}, DP(n-1, p), DP(n, p) \right)$$

$\Theta(KM)$

Return  $DP(N, p_{\text{max}})$

→ Time complexity of  $\Theta(pNKM)$

→ Space complexity of  $\Theta(Np)$

g) We will again use a 2D array, this time with parameters  $n$  and  $p$ .

$$DP = [p_{\text{max}} + 1] \times U \times N$$

For  $n$  in range  $N$ :  $) \Theta(NU)$

For  $r$  in range  $U$ :

$$DP(n, r) = \min_{k, m} \begin{cases} DP(n-1, r), \\ DP(n, r), \\ DP(n-1, r - r_{n, k, m}) + p_{n, k, m} \end{cases}$$

$\Theta(KM)$

- Time complexity of  $\Theta(UNKM)$
- Space complexity of  $\Theta(NV)$

$DP(n, r) = \min_{n' \leq n} t_q$  utilise que  
et  $DP(n, r)$  permet d'attendre

$\pi$