

# Cours 4: Programmation linéaire

- Position du problème
- Algorithme du simplexe générique
- Dualité.
- Dégénérescence et terminaison de l'algorithme

# Cours 4: Programmation linéaire

- Position du problème
- Algorithme du simplexe générique
- Dualité.
- Dégénérescence et terminaison de l'algorithme

# Un problème de programmation linéaire

**Donnée:** Une matrice réelle  $A = (a_{i,j})$  de taille  $m \times n$ ,  
un vecteur  $b = (b_1, \dots, b_m)$  de taille  $m$ ,  
un vecteur  $c = (c_1, \dots, c_n)$  de taille  $n$ .

**Problème:** Trouver un point  $x = (x_1, \dots, x_n)$  qui

- satisfait les  $m$  contraintes  $a_{i,1}x_1 + \dots + a_{i,n}x_n \leq b_i$ ,
- et maximise le produit  $c \cdot x = c_1x_1 + \dots + c_nx_n$ .

**Reformulation matricielle:** on cherche  $\max(c \cdot x \mid Ax \leq b)$ .

**Remarque:** Les  $a_{i,j}$ ,  $b_i$  et  $c_i$  peuvent être négatifs, on peut donc modéliser des contraintes  $\geq$  ou  $=$  et chercher min au lieu de max.

## Un exemple bateau: le fleuriste

**Donnée.** En stock: 50 lys, 80 roses, 80 jonquilles.

Composition des bouquets au catalogue:

- 10 lys, 10 roses, 20 jonquilles: 4 euros       $x$  bouquets
- 10 lys, 20 roses, 10 jonquilles: 5 euros       $y$  bouquets

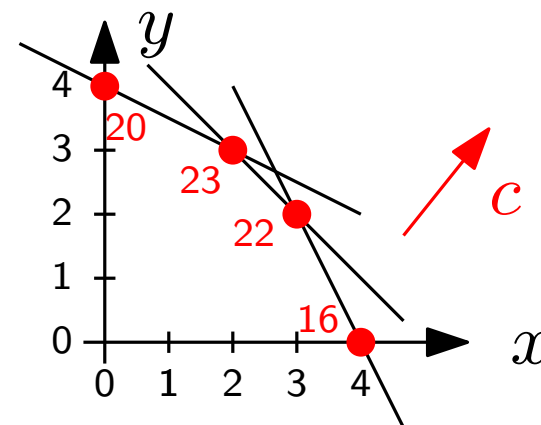
**Problème** Quels bouquets préparer si on est assuré de tout vendre ?

**Contraintes:** sur les lys:  $10x + 10y \leq 50$

sur les roses:  $10x + 20y \leq 80$

sur les jonquilles:  $20x + 10y \leq 80$

$x \geq 0$        $y \geq 0$



**Fonction économique à maximiser:**  $\max(4x + 5y)$ .

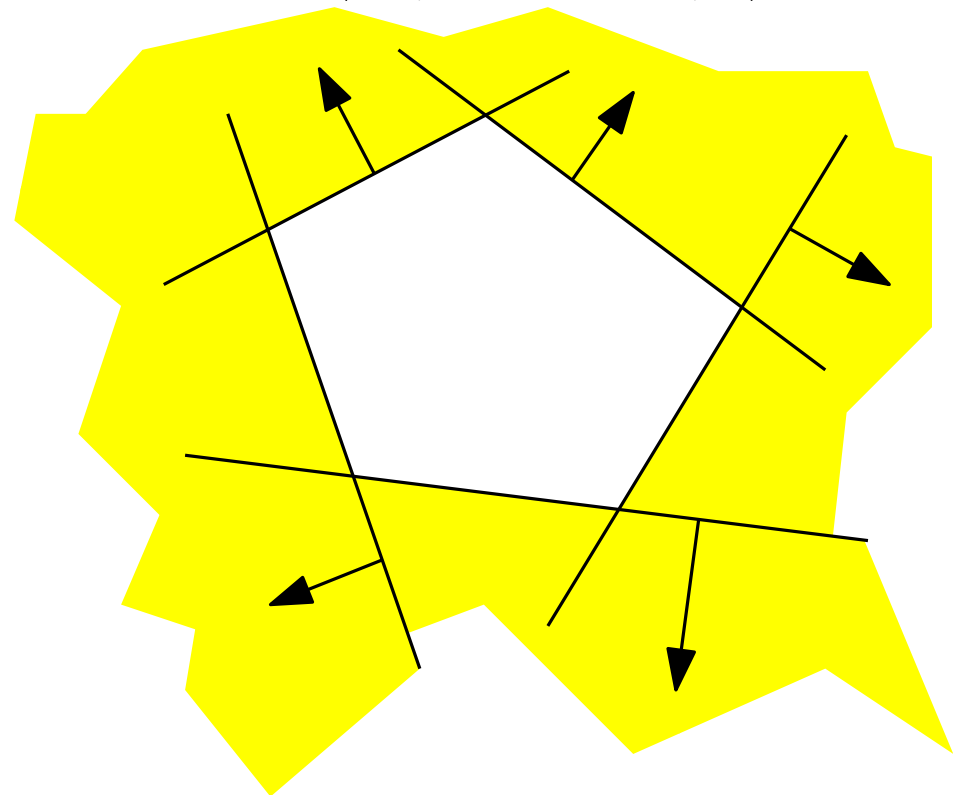
Lorsqu'on a les contraintes  $x_i \geq 0$  on peut toujours penser en termes économiques: produits finis, matières premières, bénéfice.

# Interprétation géométrique

**Donnée:** Une matrice réelle  $A = (a_{i,j})$  de taille  $m \times n$ ,  
un vecteur  $b = (b_1, \dots, b_m)$  de taille  $m$ ,  
un vecteur  $c = (c_1, \dots, c_n)$  de taille  $n$ .

Chaque équation  $a_{i,1}x_1 + \dots + a_{i,n}x_n \leq b_i$  coupe l'espace en 2,  
le long d'un l'**hyperplan** normal au vecteur  $L_i = (a_{i,1}, \dots, a_{i,n})$ .

L'ensemble des  $x$  satisfaisant  $Ax \leq b$   
forme un **polyèdre convexe**  $P$

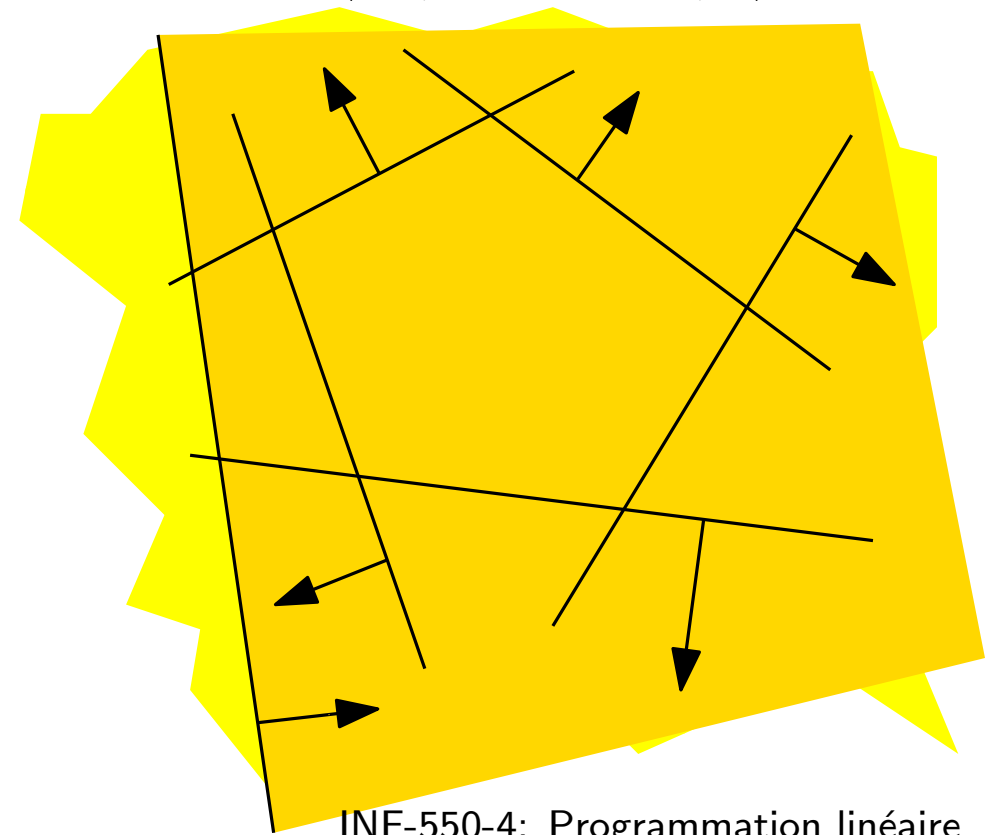


# Interprétation géométrique

**Donnée:** Une matrice réelle  $A = (a_{i,j})$  de taille  $m \times n$ ,  
un vecteur  $b = (b_1, \dots, b_m)$  de taille  $m$ ,  
un vecteur  $c = (c_1, \dots, c_n)$  de taille  $n$ .

Chaque équation  $a_{i,1}x_1 + \dots + a_{i,n}x_n \leq b_i$  coupe l'espace en 2,  
le long d'un l'**hyperplan** normal au vecteur  $L_i = (a_{i,1}, \dots, a_{i,n})$ .

L'ensemble des  $x$  satisfaisant  $Ax \leq b$   
forme un **polyèdre convexe**  $P$   
éventuellement vide

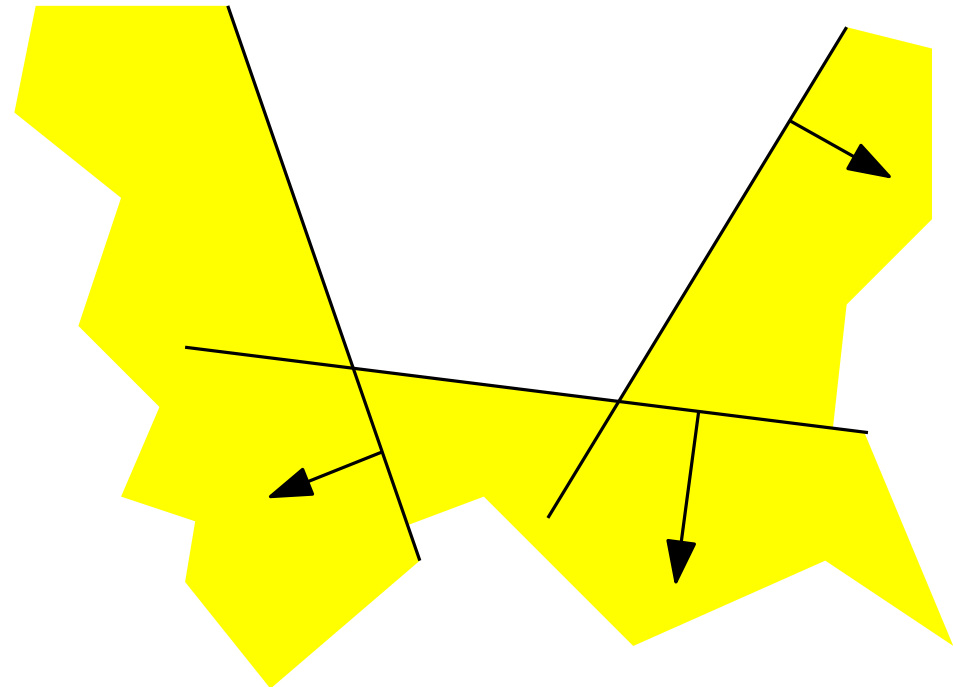


# Interprétation géométrique

**Donnée:** Une matrice réelle  $A = (a_{i,j})$  de taille  $m \times n$ ,  
un vecteur  $b = (b_1, \dots, b_m)$  de taille  $m$ ,  
un vecteur  $c = (c_1, \dots, c_n)$  de taille  $n$ .

Chaque équation  $a_{i,1}x_1 + \dots + a_{i,n}x_n \leq b_i$  coupe l'espace en 2,  
le long d'un l'**hyperplan** normal au vecteur  $L_i = (a_{i,1}, \dots, a_{i,n})$ .

L'ensemble des  $x$  satisfaisant  $Ax \leq b$   
forme un **polyèdre convexe**  $P$   
éventuellement vide ou non borné



# Interprétation géométrique

**Donnée:** Une matrice réelle  $A = (a_{i,j})$  de taille  $m \times n$ ,  
un vecteur  $b = (b_1, \dots, b_m)$  de taille  $m$ ,  
un vecteur  $c = (c_1, \dots, c_n)$  de taille  $n$ .

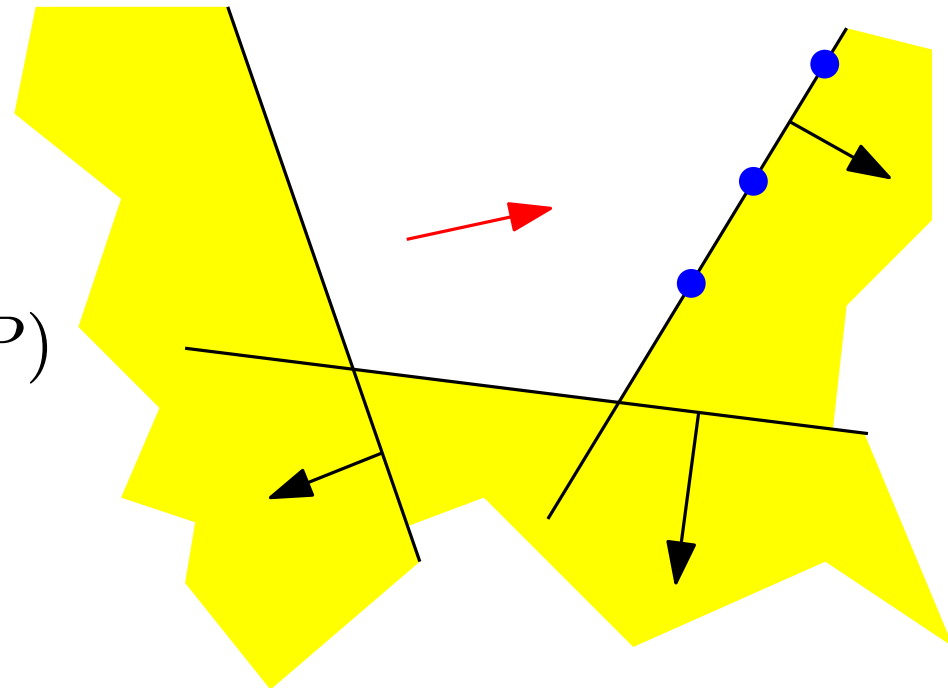
Chaque équation  $a_{i,1}x_1 + \dots + a_{i,n}x_n \leq b_i$  coupe l'espace en 2,  
le long d'un l'**hyperplan** normal au vecteur  $L_i = (a_{i,1}, \dots, a_{i,n})$ .

L'ensemble des  $x$  satisfaisant  $Ax \leq b$   
forme un **polyèdre convexe**  $P$

éventuellement vide ou non borné

Le vecteur  $c$  indique la direction dans  
laquelle on optimise:  $\max(c \cdot x \mid x \in P)$

L'optimum peut être à l'infini.



# Interprétation géométrique

**Donnée:** Une matrice réelle  $A = (a_{i,j})$  de taille  $m \times n$ ,  
un vecteur  $b = (b_1, \dots, b_m)$  de taille  $m$ ,  
un vecteur  $c = (c_1, \dots, c_n)$  de taille  $n$ .

Chaque équation  $a_{i,1}x_1 + \dots + a_{i,n}x_n \leq b_i$  coupe l'espace en 2,  
le long d'un l'**hyperplan** normal au vecteur  $L_i = (a_{i,1}, \dots, a_{i,n})$ .

L'ensemble des  $x$  satisfaisant  $Ax \leq b$   
forme un **polyèdre convexe**  $P$

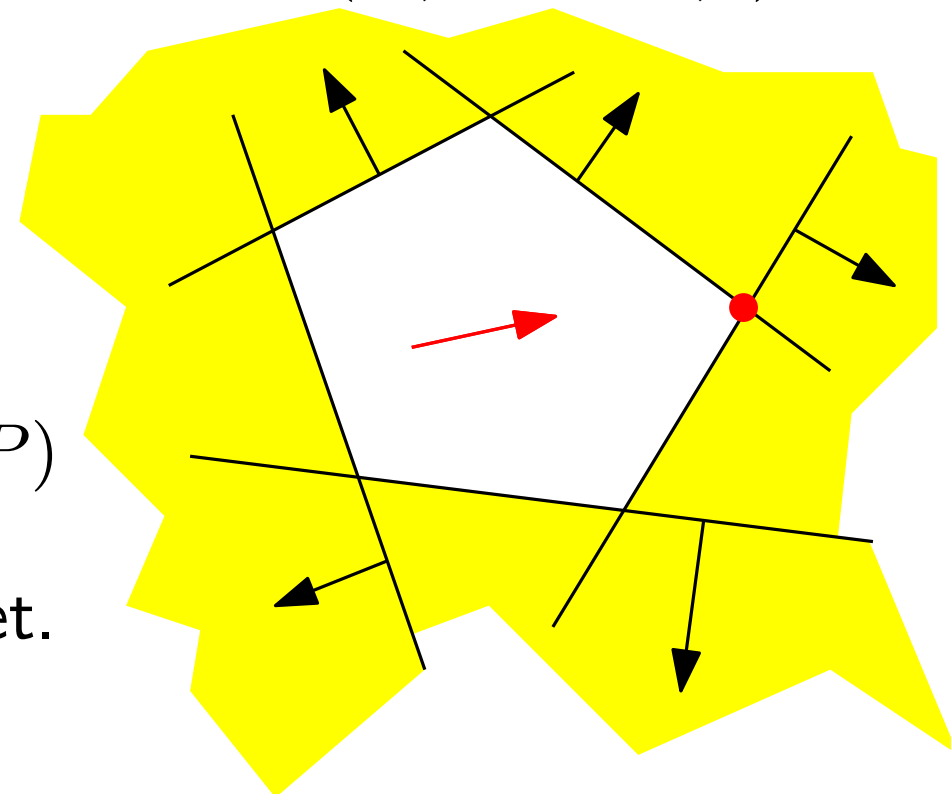
éventuellement vide ou non borné

Le vecteur  $c$  indique la direction dans  
laquelle on optimise:  $\max(c \cdot x \mid x \in P)$

L'optimum peut être à l'infini.

S'il est fini, il est atteint en un sommet.

C'est toujours le cas si  $P$  est borné.



# Interprétation géométrique

**Donnée:** Une matrice réelle  $A = (a_{i,j})$  de taille  $m \times n$ ,  
un vecteur  $b = (b_1, \dots, b_m)$  de taille  $m$ ,  
un vecteur  $c = (c_1, \dots, c_n)$  de taille  $n$ .

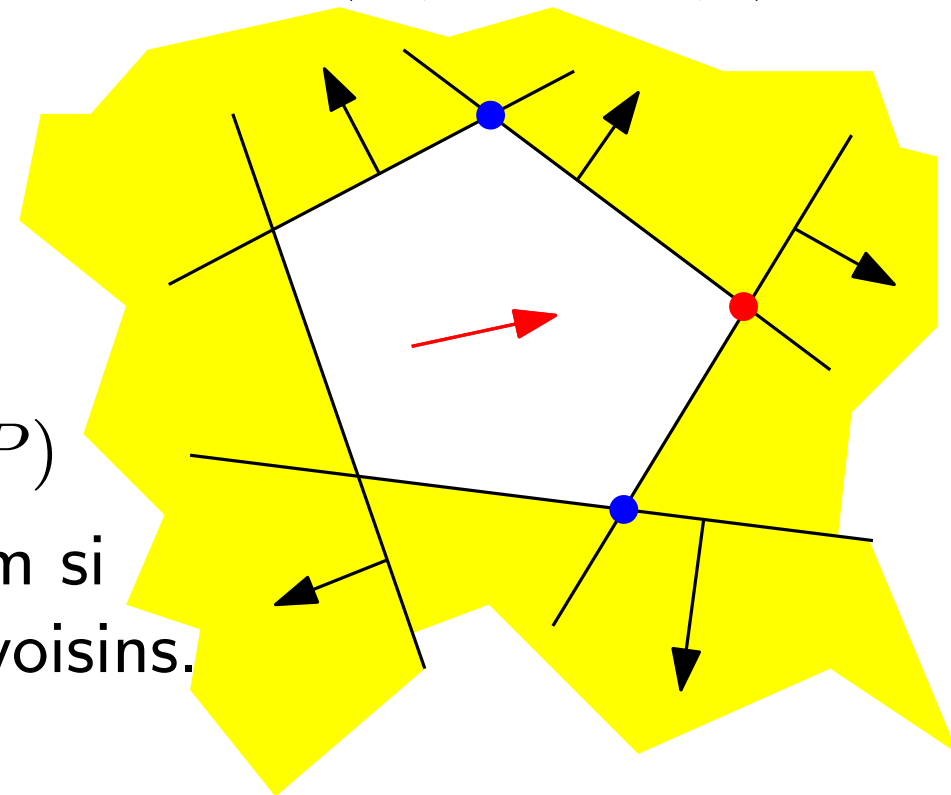
Chaque équation  $a_{i,1}x_1 + \dots + a_{i,n}x_n \leq b_i$  coupe l'espace en 2,  
le long d'un l'**hyperplan** normal au vecteur  $L_i = (a_{i,1}, \dots, a_{i,n})$ .

L'ensemble des  $x$  satisfaisant  $Ax \leq b$   
forme un **polyèdre convexe**  $P$

éventuellement vide ou non borné

Le vecteur  $c$  indique la direction dans  
laquelle on optimise:  $\max(c \cdot x \mid x \in P)$

Par convexité, un sommet est optimum si  
et seulement si il est meilleur que ses voisins.



# Cours 4: Programmation linéaire

- Position du problème
- Algorithme du simplexe générique
- Dualité.
- Dégénérescence et terminaison de l'algorithme

# Algorithme du simplexe, premier essai

- Initialiser  $x$  avec un sommet quelconque de  $P$
- Tant qu'on a pas trouvé une direction infinie où  $c \cdot x$  croit, et qu'il existe  $x'$  voisin de  $x$  avec  $c \cdot x' > c \cdot x$ , faire  $x := x'$ .

Remarques: Glouton...

Qu'est ce qu'un sommet, un voisin ?

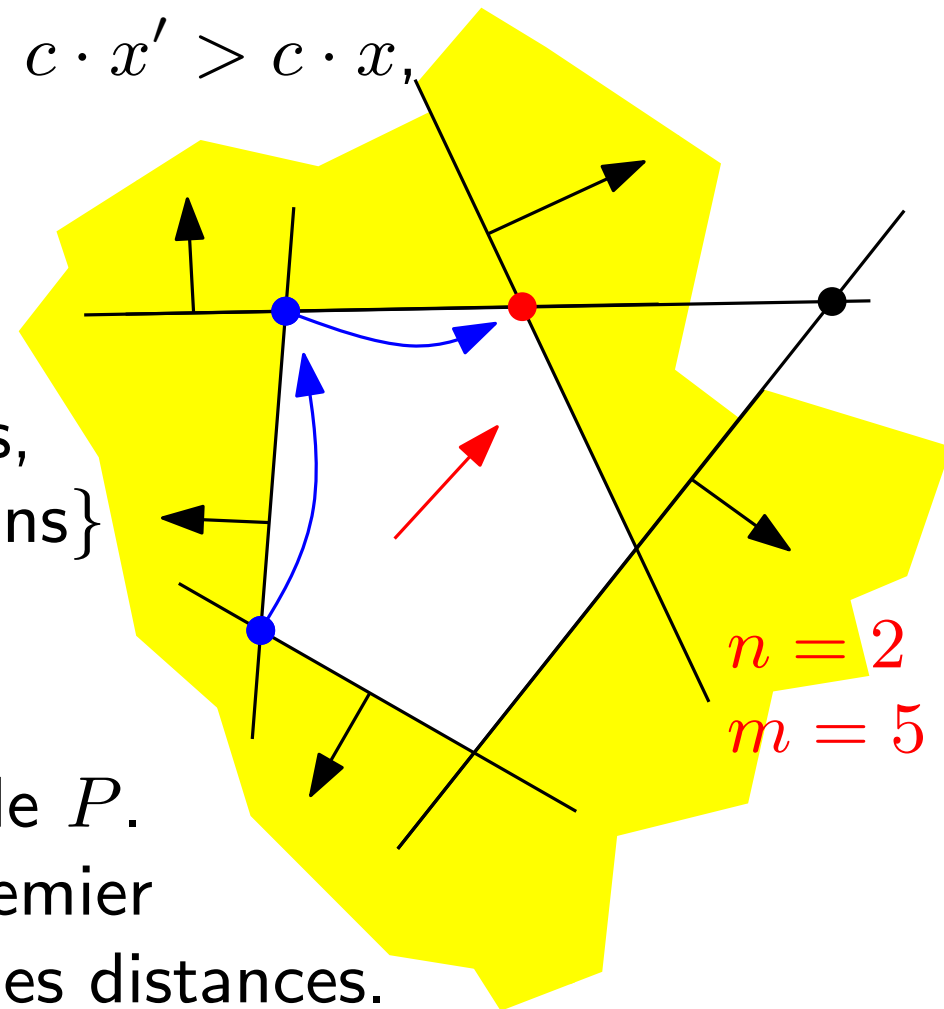
Sommet = intersection de  $n$  hyperplans,  
donné par  $I = \{\text{indices des } n \text{ équations}\}$

Voisin = partage  $n - 1$  équations

$\Rightarrow$  un sommet a  $n(m - n)$  voisins ?

On veut un voisin qui soit un sommet de  $P$ .

Le long d'une arête ( $I \setminus \{i\}$ ), seul le premier voisin rencontré est dans  $P$ : comparer les distances.



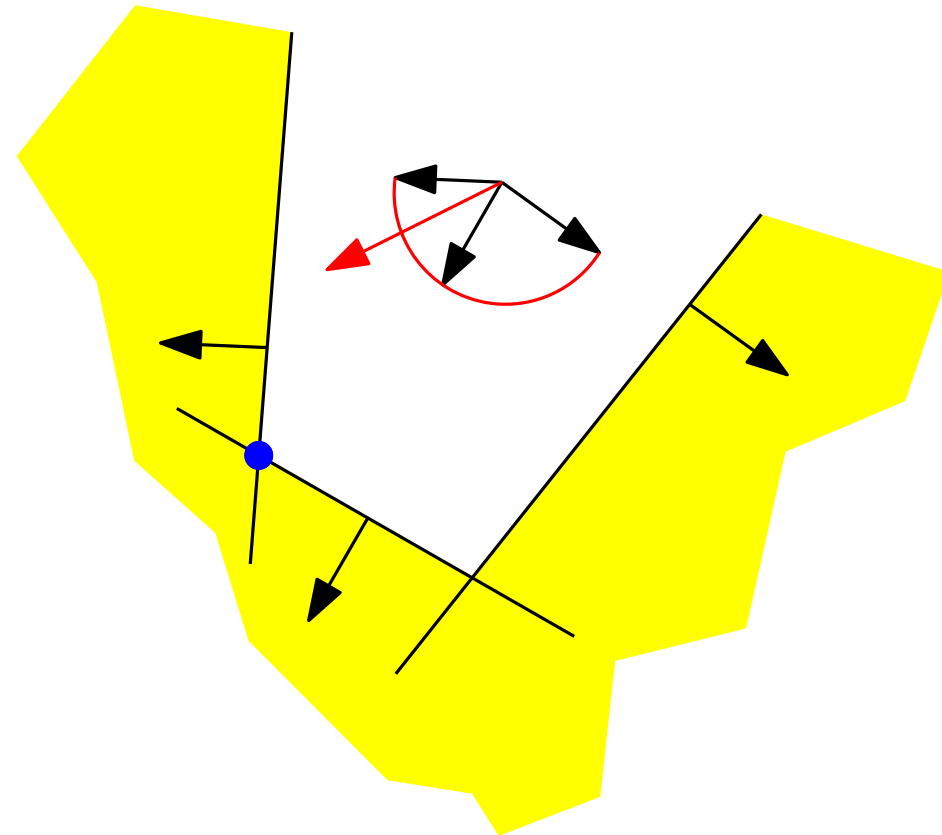
$n = 2$   
 $m = 5$

# Détecter les directions infinies où $c \cdot x$ croît.

Si le polyèdre n'est pas borné,  $c \cdot x$  peut croître indéfiniment:

**Lemme:** les 2 cas suivants s'excluent mutuellement:

- $c$  s'écrit  $c = \sum_i y_i L_i$  avec les  $y_i \geq 0 \Rightarrow$  l'optimum est borné
- il existe  $u$  tq  $c \cdot u > 0$  et  $Au \leq 0$

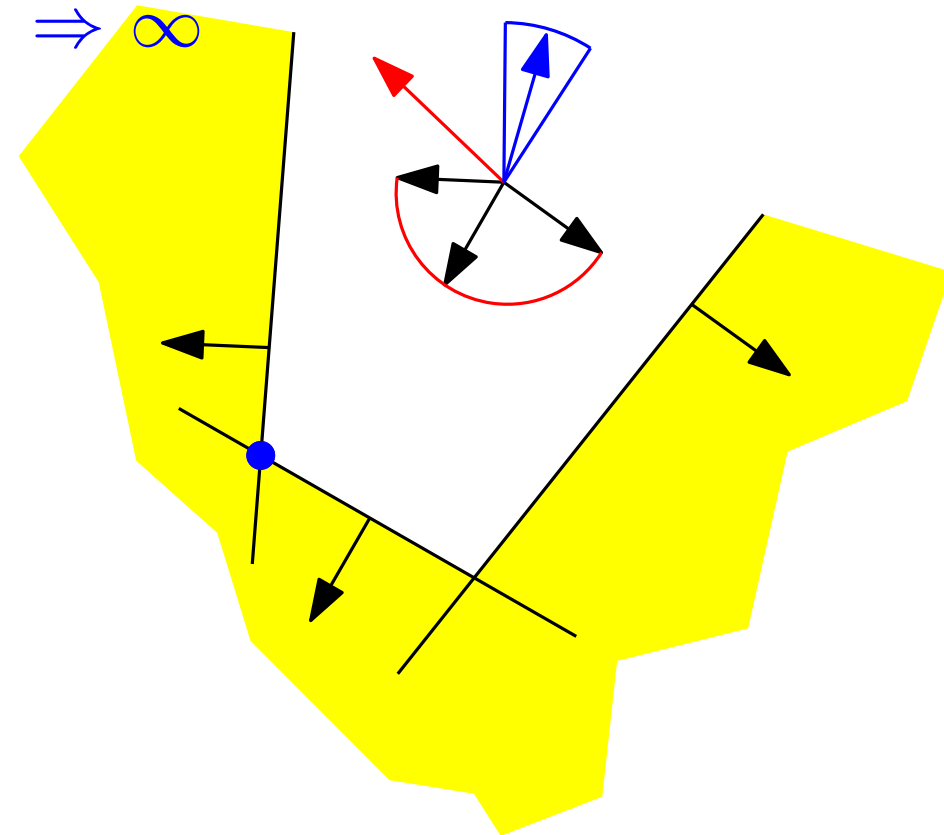


# Détecter les directions infinies où $c \cdot x$ croît.

Si le polyèdre n'est pas borné,  $c \cdot x$  peut croître indéfiniment:

**Lemme:** les 2 cas suivants s'excluent mutuellement:

- $c$  s'écrit  $c = \sum_i y_i L_i$  avec les  $y_i \geq 0 \Rightarrow$  l'optimum est borné
- il existe  $u$  tq  $c \cdot u > 0$  et  $Au \leq 0 \Rightarrow \infty$



# Détecter les directions infinies où $c \cdot x$ croît.

Si le polyèdre n'est pas borné,  $c \cdot x$  peut croître indéfiniment:

**Lemme:** les 2 cas suivants s'excluent mutuellement:

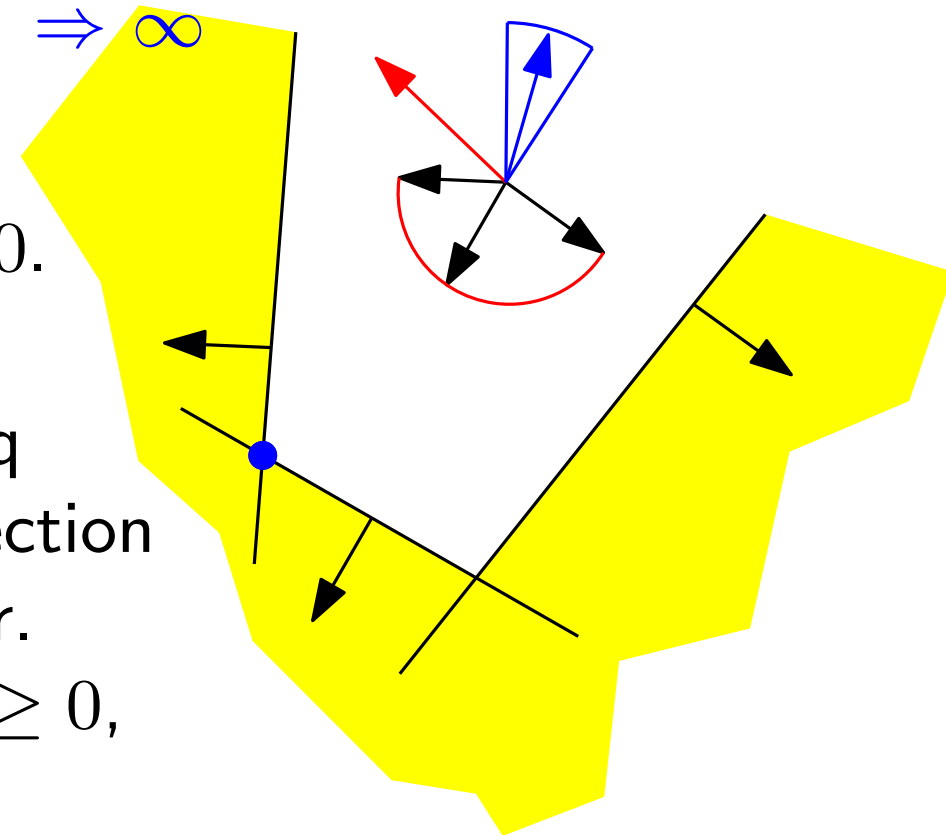
- $c$  s'écrit  $c = \sum_i y_i L_i$  avec les  $y_i \geq 0 \Rightarrow$  l'optimum est borné
- il existe  $u$  tq  $c \cdot u > 0$  et  $Au \leq 0 \Rightarrow \infty$

En effet:  $c = yA \Rightarrow cu = yAu$

donc si  $y \geq 0$  et  $Au \leq 0$ , on a  $cu \leq 0$ .

**Application:** si au cours de l'exécution on rencontre une arête de direction  $u$  tq  $Au \leq 0$  et  $cu > 0$ , on a trouvé une direction de croissance infinie et on peut s'arrêter.

Si au point  $x_I$ ,  $c = \sum_{i \in I} y_i L_i$  avec  $y_i \geq 0$ , on a trouvé l'optimum.



# Algorithme du simplexe, version *générique*

Soit  $x_I$  le sommet courant, solutions des  $n$  équations  $(L_k x = b_k)_{k \in I}$

- Exprimer  $c$  dans la base  $\{L_k\}_{k \in I}$ :  $c = \sum_{k \in I} y_k L_k$ .

Si  $y_k \geq 0$  pour tout  $k \in I$ , on a trouvé l'optimum.

- Sinon on choisit le plus petit  $i \in I$  tq  $y_i < 0$  et on considère l'arête définie par les équations de  $I' = I \setminus \{i\}$ . Soit  $u$  son vecteur directeur tq  $L_i u = -1$  (c'est une colonne de  $(A_I)^{-1}$ ).

- Calculer la vitesse vers  $L_j$  quand on suit  $u$ :  $v_j = L_j \cdot u$ .

si  $v_j \leq 0$  on s'éloigne de l'hyperplan  $L_j$ : si  $\forall j$ ,  $\max = \infty$ .

- Parmi les voisins sur l'arête, prendre celui qui est dans  $P$ .

C'est celui qui est le plus proche parmi ceux du bon côté:  
donné par un des  $j$  tels que  $v_j > 0$  qui minimisent  $\frac{b_j - L_j x_I}{v_j}$ .

faire  $I = I \setminus \{i\} \cup \{j\}$ , recalculer  $x_I$  et reprendre.

# Algorithme du simplexe, analyse

Soit  $x_I$  le sommet courant, solutions des  $n$  équations  $(L_k x = b_k)_{k \in I}$

Si on avance vers le sommet  $x_J$ ,  $J = I \setminus \{i\} \cup \{j\}$ , alors:

- $c = \sum_{k \in I} y_k L_k$ , avec  $y_i \geq 0$  pour  $k < i$ ,  $y_i < 0$ .

Comparons l'objectif  $c \cdot x_I$  avec  $c \cdot x_J$ :

- $c \cdot x_J = c \cdot x_I + c \cdot \frac{b_j - L_j x_I}{v_j} u = c \cdot x_I + \frac{b_j - L_j x_I}{v_j} c \cdot u$

or  $c \cdot u = \sum_{k \in I} y_k L_k \cdot u = -y_i > 0$  puisque  $u$  est dans les hyperplans  $L_k$ ,  $k \neq i$  et  $L_i \cdot u = -1$ .

$\Rightarrow$  comme prévu  $x_J$  améliore l'objectif... **sauf si  $b_j - L_j x = 0$  !**

Si plus de  $n$  hyperplans  $L_k$  passent par  $x_I$ , certaines arêtes dégènèrent et on peut avoir  $x_J = x_I$ , l'algorithme risque de boucler.

**Une solution:** perturber les  $b_i$  pour éliminer les dégénérescences.

# Algorithme du simplexe, initialisation

Pour démarrer l'algorithme il faut avoir un sommet de  $P$ .

On cherche:  $\operatorname{argmax}(cx \mid Ax \leq b)$

On peut toujours poser  $x_i = y_i - z_i$  (le nombre de variables double) et maximiser  $(c, -c) \cdot \begin{pmatrix} y \\ z \end{pmatrix}$  pour  $(A, -A) \begin{pmatrix} y \\ z \end{pmatrix} \leq b$  et  $\begin{pmatrix} y \\ z \end{pmatrix} \geq 0$ .

On cherche donc:  $\operatorname{argmax}(cx \mid Ax \leq b, x \geq 0)$

Si les  $b_i$  sont positifs,  $(0, \dots, 0)$  est notre sommet initial.

Sinon on cherche un sommet de  $P = \{x \mid Ax \leq b, x \geq 0\}$

On fait cela en cherchant parmi les  $(x_1, \dots, x_n, t)$  tq

$$a_{i,1}x_1 + \dots + a_{i,n}x_n - t \leq b_i \text{ pour tout } i,$$

un point qui minimise  $t$ .

On connaît un sommet initial  $(0, \dots, 0, T)$  pour ce problème linéaire.

Il existe (et on obtient) un sommet de  $P$  ssi  $t = 0$  dans sa solution.

# Cours 4: Programmation linéaire

- Position du problème
- Algorithme du simplexe générique
- Dualité.
- Dégénérescence et terminaison de l'algorithme

# Primal/dual et théorème de dualité

Théorème (cas inégalités+positivité). Si l'un des 2 problèmes suivant

$$(\mathcal{P}) \quad \max(cx \mid Ax \leq b, x \geq 0)$$

$$(\mathcal{D}) \quad \min(yb \mid yA \geq c, y \geq 0)$$

admet une solution finie, alors l'autre aussi et on a

$$\min(yb \mid yA \geq c, y \geq 0) = \max(cx \mid Ax \leq b, x \geq 0)$$

Interprétation. Retour au fleuriste.

	Primal	Le théorème dit	Dual
×	$\max(4x + 5x')$	qu'on peut obtenir	$\min(50y_1 + 80y_2 + 80y_3)$
$y_1$	$10x + 10x' \leq 50$	une borne optimale	$10y_1 + 10y_2 + 20y_3 \geq 4$
$y_2$	$10x + 20x' \leq 80$	par multiplicateur.	$10y_1 + 20y_2 + 10y_3 \geq 5$
$y_3$	$20x + 10x' \leq 80$	Si on augmente la	$y_1 \geq 0, y_2 \geq 0, y_3 \geq 0.$
	$x \geq 0, x' \geq 0$	denrée $i$ de $\Delta_i$ , le gain	
		augmente de $\Delta_i y_i$ .	

$$\Rightarrow 4x + 5x' \leq (10y_1 + 10y_2 + 20y_3)x + (10y_1 + 20y_2 + 10y_3)x' \leq 50y_1 + 80y_2 + 80y_3$$

# Primal/dual et théorème de dualité

Primal / Dual, cas général. Le théorème s'applique aux paires:

Primal

$$\max(c_1x_1 + \dots + c_nx_n)$$

$$a_{i1}x_1 + \dots + a_{in}x_n \leq b_i \text{ pour } i \in I$$

$$a_{i1}x_1 + \dots + a_{in}x_n = b_i \text{ pour } i \in E$$

$$x_j \geq 0 \text{ pour } j \in P.$$

Dual

$$\min(b_1y_1 + \dots + b_my_m)$$

$$a_{1j}y_1 + \dots + a_{mj}y_m \geq c_j \text{ pour } j \in P$$

$$a_{1j}y_1 + \dots + a_{mj}y_m = c_j \text{ pour } j \in N$$

$$y_i \geq 0 \text{ pour } i \in I.$$

$$\text{où } m = |I| + |E| \text{ et } n = |P| + |N|$$

Les inégalité donnent dans le dual des variables contraintes à être positives, les égalités des variables quelconques (penser à l'interprétation des variables duales comme multiplicateurs).

En TD: le théorème de dualité  $\Rightarrow$  maxflow = mincut

# Cours 4: Programmation linéaire

- Position du problème
- Algorithme du simplexe générique
- Dualité.
- Dégénérescence et terminaison de l'algorithme

# Dégénérescence et risque de bouclage

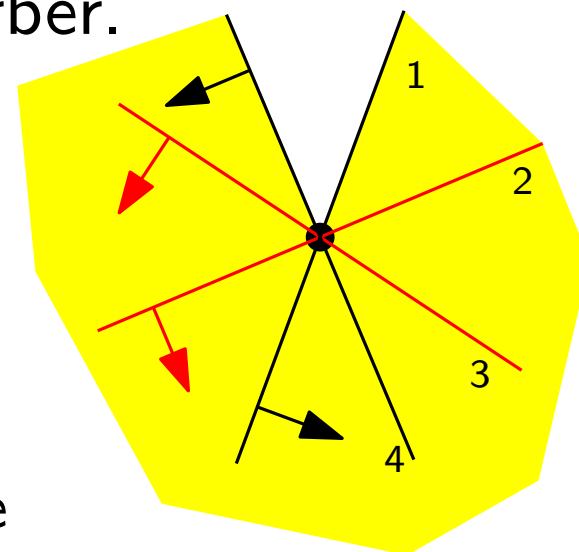
**Dégénérescence:** un sommet appartient à plus de  $n$  hyperplans.

Il n'est pas forcément acceptable de perturber.

Parmi les voisins de  $x_{\{2,3\}}$  il y a

$x_{\{1,3\}}$ ,  $x_{\{1,4\}}$ ,  $x_{\{1,2\}}$ ,  $x_{\{2,4\}}$ .

Ce sont même ses *plus proches voisins*...



En dimension supérieure il peut être nécessaire de changer plusieurs lignes pour trouver un vrai voisin dans  $P$ .

Il peut y avoir un nombre exponentiel de faux voisins.

Notre algorithme du simplexe prend **un** voisin le plus proche: il faut bien le choisir pour ne pas risquer de boucler indéfiniment en changeant  $I$  sans

changer  $x_I$ :  $x_{\{2,3\}} \rightarrow x_{\{1,3\}} \rightarrow x_{\{1,2\}} \rightarrow x_{\{2,3\}}$ .

# Algorithme du simplexe, version finale

Soit  $x_I$  le sommet courant, solutions des  $n$  équations  $(L_k x = b_k)_{k \in I}$

- Exprimer  $c$  dans la base  $\{L_k\}_{k \in I}$ :  $c = \sum_{k \in I} y_k L_k$ .

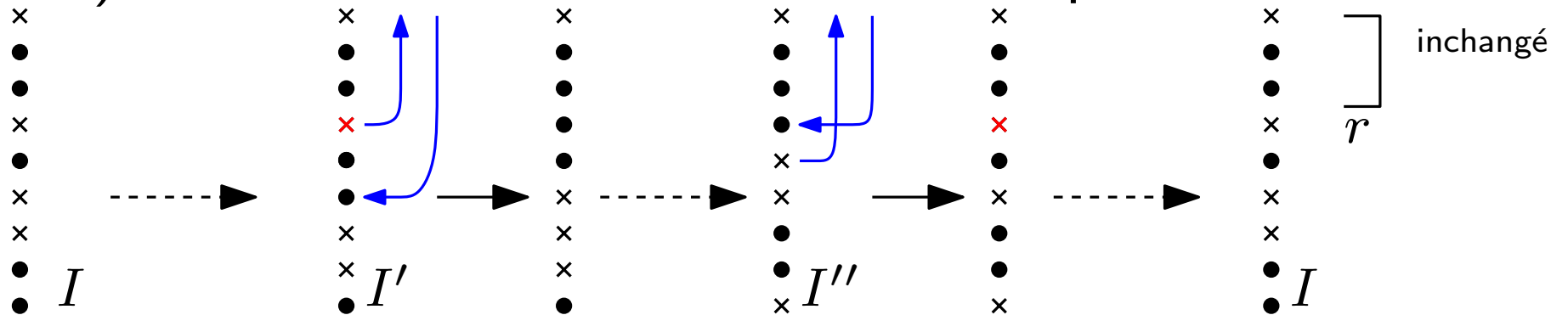
Si  $y_k \geq 0$  pour tout  $k \in I$ , on a trouvé l'optimum.

- Sinon on choisit le plus petit  $i \in I$  tq  $y_i < 0$  et on considère l'arête définie par les équations de  $I' = I \setminus \{i\}$ . Soit  $u$  son vecteur directeur tq  $L_i u = -1$  (c'est une colonne de  $(A_I)^{-1}$ ).
- Calculer la vitesse vers  $L_j$  quand on suit  $u$ :  $v_j = L_j \cdot u$ .  
si  $v_j \leq 0$  on s'éloigne de l'hyperplan  $L_j$ : si  $\forall j$ ,  $\max = \infty$ .
- Parmi les voisins sur l'arête, prendre celui qui est dans  $P$ , c'est celui qui est le plus proche parmi ceux du bon côté:  
le **plus petit parmi les**  $j$  tels que  $v_j > 0$  qui minimisent  $\frac{b_j - L_j x_I}{v_j}$ .

faire  $I = I \setminus \{i\} \cup \{j\}$ , recalculer  $x_I$  et reprendre.

# Algorithme du simplexe, preuve de terminaison

Si l'algo boucle c'est qu'on est revenu au même  $I$ , sans changer  $c \cdot x_I$  ( $c \cdot x_I$  croît). On observe l'évolution des indices présents dans  $I$ .



Soit  $r$  le plus grand indice qui est entre ou sort durant la boucle,  $I'$  et  $I''$  les ensembles d'indices à des temps d'entrée et de sortie de  $r$ .

Comme  $r$  sort de  $I'$ :  $c = \sum_{i \in I'} y_i L_i$ , avec  $y_i \geq 0$  pour  $i < r$ , et  $y_r < 0$ .

Soit  $u$  la direction de l'arête sélectionnée lors du traitement de  $I''$ : alors  $L_i u = 0$  pour  $i \in I'' \setminus \{r\}$  et, comme  $r$  entre dans  $I''$ , on a pour  $j < r$ ,  $L_j u < 0$  dès que  $b_j - L_j u = 0$ , ce qui est le cas pour  $j \in I' \setminus I''$ .

D'où  $cu = \sum_{i \in I'} y_i L_i u = \sum_{i < r, i \in I'} y_i L_i u + y_r L_r u < 0$ .

Ceci contredit le fait que suivant le  $u$  choisi on améliore l'objectif: ( $cu > 0$ ).

# Algorithme du simplexe, complexité

Puisque le nombre de voisins d'un sommet est  $n(m - 1)$  au plus, chaque étape a un coût polynomial.

Le nombre de sommets peut être exponentiel (jusqu'à  $\binom{m}{n}$ ) en  $m$  et  $n$ , et il existe des exemples pour lesquels le simplexe visite effectivement un nombre exponentiel de sommets... (Worst case analysis).

On constate que cela ne se produit quasiment jamais en pratique...

On peut montrer qu'en moyenne sur des problèmes aléatoires l'algorithme est polynomial (Average case analysis).

Mieux on peut montrer que si on perturbe aléatoirement des données arbitraires, l'algorithme reste polynomial (Smooth analysis).

# Programmation linéaire, algorithmes polynomiaux.

Le simplexe n'est pas polynomial mais presque...

De fait il existe des algorithmes polynomiaux pour la programmation linéaire: notamment les méthodes de l'ellipsoïde et du point intérieur.

**Attention toutefois:** il y a des algorithmes **polynomiaux** si on cherche les optimaux à **coordonnées dans  $\mathbb{R}$** .

On verra que, contrairement à ce qui se passe pour les problèmes de flots, le **problème devient dur** si on veut des **solutions entières**.

# Cours 4: Programmation linéaire

- Position du problème
- Algorithme du simplexe générique
- Dualité.
- Dégénérescence et terminaison de l'algorithme

**Retenir:** - Il existe d'excellentes boîtes noires pour résoudre les LP.  
- Modélisation par LP (en PC). Thm Primal-Dual.