IFT 2505
Programmation Linéaire
Introduction

Fabian Bastin
DIRO
Université de Montréal

Objectifs du cours

1. Bases de la programmation mathématique :
   - objectif, contraintes ;
   - solution(s) réalisable(s).
   - solution(s) optimale(s), valeur optimale ;
2. Comprendre les principes de base de la programmation linéaire (PL), et de ses méthodes de résolution. La méthode du simplexe sera étudiée avec un niveau élevé de détails.
3. Comprendre la notion de dualité.
4. Principes de base des méthodes de points intérieurs.
5. Introduction à la programmation en nombres entiers.
6. Aspects numériques de la PL (utilisation de Julia).

Ouvrages de références

- Luenberger et Ye, Linear and Nonlinear Programming, 4e édition, Springer, 2016, https://link.springer.com/book/10.1007%2F978-3-319-18842-3
- Chan, Batson, Dang, Applied Integer Programming : Modeling and Solution, Wiley, 2011, https://onlinelibrary.wiley.com/doi/book/10.1002/9781118166000

Modalités

Évaluations :

- 5 devoirs, 10% chaque ;
- Examen intra : 20%, une page de notes autorisée ;
- Examen final : 30%, deux pages de notes autorisées.

Le règlement sur le plagiat sera de stricte application !

Démonstrateur : Jean Laprés-Chartrand

- Notebooks : `https://github.com/fbastin/optim` ;
- Slides : `https://github.com/fbastin/LP_Introduction` ;
- Canal discord : `https://discord.gg/Z4ZU6P8K`.

PL : forme standard

$$
\begin{aligned}
\min_{\boldsymbol{x}} \quad & z = c_1x_1 + c_2x_2 + \ldots + c_nx_n \\
\text{sujet à} \quad & a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = b_1 \\
& a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n = b_2 \\
& \qquad \vdots \qquad\quad \vdots \\
& a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n = b_m \\
& x_1 \geq 0, \;\; x_2 \geq 0, \ldots, x_n \geq 0
\end{aligned}
$$

Ecriture matricielle

$$\begin{aligned} \min_{\boldsymbol{x}} \;\; & z = \boldsymbol{c}^T x \\ \text{sujet à} \;\; & \boldsymbol{A}x = \boldsymbol{b} \\ & \boldsymbol{x} \geq 0 \end{aligned}$$

Retour à la forme standard

Variables d'écart

$$a_{j1}x_1 + a_{j2}x_2 + \ldots + a_{jn}x_n \leq b_j$$
$$\rightarrow a_{j1}x_1 + a_{j2}x_2 + \ldots + a_{jn}x_n + s_j = b_j$$

Variables de surplus

$$a_{j1}x_1 + a_{j2}x_2 + \ldots + a_{jn}x_n \geq b_j$$
$$\rightarrow a_{j1}x_1 + a_{j2}x_2 + \ldots + a_{jn}x_n - u_j = b_j$$

Retour à la forme standard (suite)

Variables libres

x_j non contraint : $\cancel{x_j \geq 0}$

Première approche : décomposition

$$x_j = s_j - u_j, \quad s_j \geq 0,\ u_j \geq 0.$$

Deuxième approache : substitution

Supposons $a_{kj} \neq 0$

$$x_j = \frac{b_k}{a_{kj}} - \frac{a_{k1}}{a_{kj}} x_1 - \frac{a_{k2}}{a_{kj}} x_2 - \ldots - \frac{a_{k,j-1}}{a_{kj}} x_{j-1} - \frac{a_{k,j+1}}{a_{kj}} x_{j+1} - \ldots - \frac{a_{kn}}{a_{kj}} x_n.$$

On remplace dans les autres contraintes.

Maximisation vs minimisation

Le problème

$$\begin{aligned} \max_{\boldsymbol{x}} \ & \boldsymbol{c}^T x \\ \text{sujet à } & \boldsymbol{A}x = \boldsymbol{b} \\ & \boldsymbol{x} \geq 0 \end{aligned}$$

est équivalent à

$$\begin{aligned} -\min_{\boldsymbol{x}} \ & -\boldsymbol{c}^T x \\ \text{sujet à } & \boldsymbol{A}x = \boldsymbol{b} \\ & \boldsymbol{x} \geq 0 \end{aligned}$$

Réalisabilité – optimalité

$\boldsymbol{x}$ est une solution réalisable si $\boldsymbol{Ax} = \boldsymbol{b}$ et $\boldsymbol{x} \geq 0$.

$\boldsymbol{x}$ est une solution optimale si $\boldsymbol{x}$ est réalisable et il n'existe pas de de $\boldsymbol{x}'$ réalisable tel que $\boldsymbol{c}^T\boldsymbol{x}' < \boldsymbol{c}^T\boldsymbol{x}$.

Supposons $m \leq n$ et $\text{rang}(\boldsymbol{A}) = m$. Sans perte de généralité, supposons les m premières colonnes de $\boldsymbol{A}$ indépendantes, et formons

$$\boldsymbol{A} = \begin{pmatrix} \boldsymbol{B} & \boldsymbol{D} \end{pmatrix}$$

Solution de base : $\boldsymbol{x} = (\boldsymbol{x}_b \; 0)$, avec $\boldsymbol{B}\boldsymbol{x}_b = \boldsymbol{b}$.

Exemple (Wyndor Glass)

La compagnie Wyndor Glass Co. produit des produits verriers de haute qualité, incluant des fenêtres et des portes vitrées. Elle dispose à cette fin de trois usines (usine 1, usine 2, usine 3), qui ont chacune une capacité de production limitée. Les châssis en aluminium et les matériaux sont produits dans l'usine 1, les châssis en bois sont fabriqués dans l'usine 2, et l'usine 3 produit le verre et assemble les produits. La compagnie a décidé de mettre en place de ligne de production :

- *produit 1 : une porte vitrée avec un châssis d'aluminium ;*
- *produit 2 : une fenêtre double-vitrage avec châssis en bois.*

Un lot de 20 unités donne lieu à un profit de $3000 et $5000, respectivement pour le produit 1 et le produit 2.

Exemple (Wyndor Glass)

	Produit 1 Tps de prod. (h)	Produit 2 Tps de prod. (h)	Capacité de production (h)
Usine 1	*1*	*0*	*4*
Usine 2	*0*	*2*	*12*
Usine 3	*3*	*2*	*18*

Chaque lot d'un produit est le résultat combiné de la production dans les trois usines. Nous souhaitons déterminer le taux de production pour chaque produit (nombre de lots par semaine) de façon à maximiser le profit total.
Les variables de décision sont

- x_1*, le nombre de lots du produit 1 ;*
- x_2*, le nombre de lots du produit 2.*

Le fonction objectif est le profit total, qui vaut $3x_1 + 5x_2$*, en l'exprimant en miller de dollars. Nous voulons maximiser ce profit.*

Exemple (Wyndor Glass)

En résumé, nous avons le problème d'optimisation suivant :

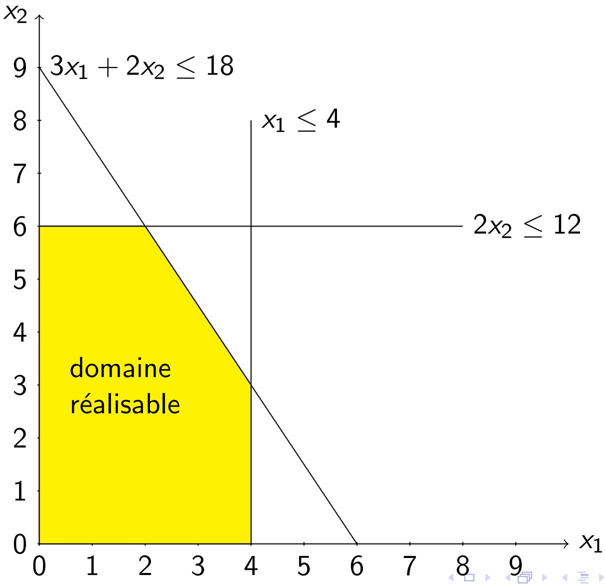
$$\begin{aligned}
\max_{x}\ & z = 3x_1 + 5x_2 & & \\
\text{t.q. } & x_1 & & \leq 4 \ \textit{(usine 1)} \\
& 2x_2 & & \leq 12 \ \textit{(usine 2)} \\
& 3x_1 + 2x_2 & & \leq 18 \ \textit{(usine 3)} \\
& x_1 \geq 0,\ x_2 \geq 0 & \textit{(non-négativité)} &
\end{aligned}$$

Dans l'exemple

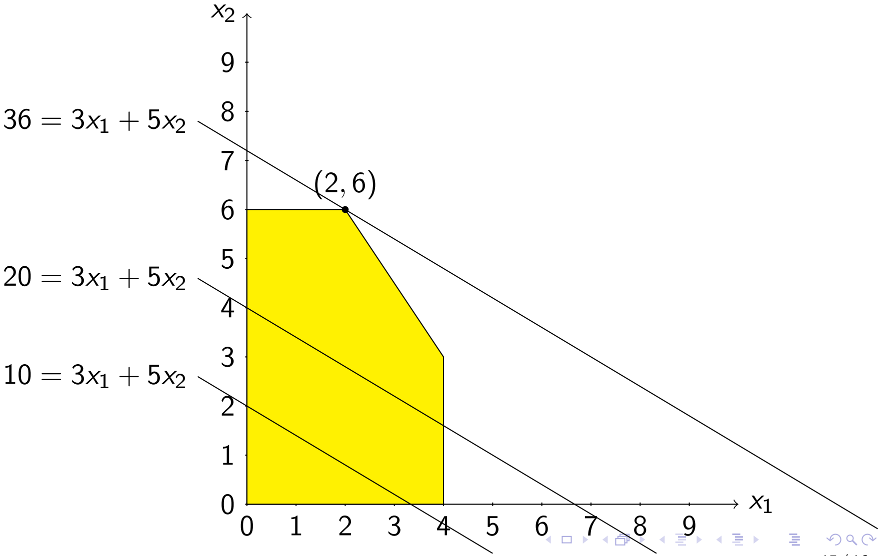
$$z = 3x_1 + 5x_2 \Leftrightarrow x_2 = -\frac{3}{5}x_1 + \frac{1}{5}z.$$

L'ordonnée à l'origine, dépendant de la valeur de z, est $\frac{1}{5}z$, et la pente vaut $-\frac{3}{5}$. Maximiser revient à augmenter z.

Solution graphique



Solution graphique



JuMP

JuMP : Julia for Mathematical Optimization

```
using JuMP
using Clp

m = Model(with_optimizer(Clp.Optimizer))

@variable(m, 0 <= x <= 4)
@variable(m, 0 <= y <= 6)

@constraint(m, 3x + 2y <= 18.0)

@objective(m, Max, 3x+5y)

status = optimize!(m)
```