

# Modèles de Localisation et Applications (MLA)

*Daniel Porumbel*

daniel.porumbel@cnam.fr

cedric.cnam.fr/~porumbed/mla/

# Rappel : le problème de localisation simple

- $x_{ij} = 1$  : le client  $j \in \mathcal{N}$  est affecté au site  $i \in \mathcal{M}$
- $y_i = 1$  : le site  $i \in \mathcal{M}$  est ouvert
- J'ai inversé  $i$  avec  $j$  par rapport à la formulation que vous avez déjà vue en MLA (avec Mme. Elloumi)

$$\begin{aligned} \min \quad & \sum_{i \in \mathcal{M}} f_i y_i + \sum_{j \in \mathcal{N}} \sum_{i \in \mathcal{M}} c_{ij} x_{ij} \\ & \sum_{i \in \mathcal{M}} x_{ij} = 1 && \forall \text{client } j \in \mathcal{N} \\ & x_{ij} \leq y_i && \forall \text{site } i \in \mathcal{M}, \forall \text{client } j \in \mathcal{N} \\ & x_{ij} \geq 0, y_i \in \{0, 1\} \end{aligned}$$

# Le problème de localisation simple

Version ultra-simple « cas d'école »

- un seul client (pas besoin d'indice  $j$ ) à affecter à un seul site

$$\begin{aligned} \min \quad & \sum_{i \in \mathcal{M}} \overbrace{f_i}^{\text{coût d'ouverture du site } i} y_i + \sum_{i \in \mathcal{M}} \underbrace{c_i}_{\text{coût d'affectation au site } i} x_i \\ & \sum_{i \in \mathcal{M}} x_i = 1 \\ & x_i \leq y_i \quad \forall i \in M \\ & x_i \geq 0, y_i \in \{0, 1\} \end{aligned}$$

Calculer la solution optimale

f :	7	2	2	7	7
c :	60	5	4	3	2

# Le problème de localisation simple

Version ultra-simple « cas d'école »

- un seul client (pas besoin d'indice  $j$ ) à affecter à un seul site

$$\begin{aligned} \min \quad & \sum_{i \in \mathcal{M}} \overbrace{f_i}^{\text{coût d'ouverture du site } i} y_i + \sum_{i \in \mathcal{M}} \underbrace{c_i}_{\text{coût d'affectation au site } i} x_i \\ & \sum_{i \in \mathcal{M}} x_i = 1 \\ & x_i \leq y_i \quad \forall i \in \mathcal{M} \\ & x_i \geq 0, y_i \in \{0, 1\} \end{aligned}$$

Calculer la solution optimale

f :	7	2	2	7	7
c :	60	5	4	3	2

# Le problème de localisation simple 2

Version 2 :

- Ajouter des contraintes sur  $\mathbf{y}$

$$\begin{aligned} \min \quad & \sum_{i \in \mathcal{M}} f_i y_i + \sum_{i \in \mathcal{M}} c_i x_i \\ & y_1 \geq y_2, y_1 \geq y_3 \\ & \sum_{i \in \mathcal{M}} x_i = 1 \\ & x_i \leq y_i \quad \forall i \in \mathcal{M} \\ & x_i \geq 0, y_i \in \{0, 1\} \end{aligned}$$

Calculer la solution optimale

f :	7	2	2	7	7
c :	60	5	4	3	2

# Le problème de localisation simple 2

Version 2 :

- Ajouter des contraintes sur  $\mathbf{y}$

$$\begin{aligned} \min \quad & \sum_{i \in \mathcal{M}} f_i y_i + \sum_{i \in \mathcal{M}} c_i x_i \\ & y_1 \geq y_2, y_1 \geq y_3 \\ & \sum_{i \in \mathcal{M}} x_i = 1 \\ & x_i \leq y_i \quad \forall i \in \mathcal{M} \\ & x_i \geq 0, y_i \in \{0, 1\} \end{aligned}$$

Calculer la solution optimale

f :	7	2	2	7	7
c :	60	5	4	3	2

# Le problème de localisation simple 3

Version 3 :

- On considère une **demande de  $d = 2$**

$$\begin{aligned} \min \quad & \sum_{i \in \mathcal{M}} f_i y_i + \sum_{i \in \mathcal{M}} c_i x_i \\ & y_1 \geq y_2, \quad y_1 \geq y_3 \\ & \sum_{i \in \mathcal{M}} x_i = d \quad (= 2) \\ & x_i \leq y_i \quad \forall i \in \mathcal{M} \\ & x_i \geq 0, y_i \in \{0, 1\} \end{aligned}$$

On veut regarder  $d = 2$  films à  $d$  cinéma différents ;

**y** le prix du billet à chaque cinéma

**x** le prix du trajet vers chaque cinéma

$y_1 \geq y_2$  Au cinéma 2 il faut acheter une paire de lunettes 3D qui coûte  $y_1$ .

$y_1 \geq y_3$  Idem pour cinéma 3, mais une paire suffit pour  $y_2$  et  $y_3$

# Le problème de localisation simple 3

Version 3 :

- On considère une demande de  $d = 2$

$$\begin{aligned} \min \quad & \sum_{i \in \mathcal{M}} f_i y_i + \sum_{i \in \mathcal{M}} c_i x_i \\ & y_1 \geq y_2, y_1 \geq y_3 \\ & \sum_{i \in \mathcal{M}} x_i = d \quad (= 2) \\ & x_i \leq y_i \quad \forall i \in \mathcal{M} \\ & x_i \geq 0, y_i \in \{0, 1\} \end{aligned}$$

Calculer la solution optimale

f :	7	2	2	7	7
c :	$\infty$	5	4	3	2



# Le problème de localisation simple 3

Version 3 :

- On considère une demande de  $d = 2$

$$\begin{aligned} \min \quad & \sum_{i \in \mathcal{M}} f_i y_i + \sum_{i \in \mathcal{M}} c_i x_i \\ & y_1 \geq y_2, y_1 \geq y_3 \\ & \sum_{i \in \mathcal{M}} x_i = d \quad (= 2) \\ & x_i \leq y_i \quad \forall i \in \mathcal{M} \\ & x_i \geq 0, y_i \in \{0, 1\} \end{aligned}$$

Calculer la solution optimale

f :	7	1	1	7	7
c :	$\infty$	5	4	3	2

# Reformulation Benders 1

$$\begin{aligned} \min \quad & \sum_{i \in \mathcal{M}} f_i y_i + w \\ & y_1 \geq y_2, \quad y_1 \geq y_3 \\ & w \geq F(\mathbf{y}) = \min_{\mathbf{x}} \sum_{i \in \mathcal{M}} c_i x_i \\ & \quad \sum_{i \in \mathcal{M}} x_i = d \\ & \quad x_i \leq y_i \quad \forall i \in \mathcal{M} \\ & w \geq 0, y_i \in \{0, 1\}, x_i \geq 0 \end{aligned}$$

On va dualiser  $F(\mathbf{y})$  en utilisant les variables duales  $b$  et  $\mathbf{v}$ .

# Reformulation Benders 1

$$\begin{aligned} \min \quad & \sum_{i \in \mathcal{M}} f_i y_i + w \\ & y_1 \geq y_2, \quad y_1 \geq y_3 \\ & w \geq F(\mathbf{y}) = \min_{\mathbf{x}} \sum_{i \in \mathcal{M}} c_i x_i \\ & \sum_{i \in \mathcal{M}} x_i = d \quad (b) \\ & x_i \leq y_i \quad \forall i \in \mathcal{M} \quad (v_i) \\ & w \geq 0, y_i \in \{0, 1\}, x_i \geq 0 \end{aligned}$$

On va dualiser  $F(\mathbf{y})$  en utilisant les variables duales  $b$  et  $\mathbf{v}$ .

# Reformulation Benders 2

$$\min \sum_{i \in \mathcal{M}} f_i y_i + w$$

$$y_1 \geq y_2, y_1 \geq y_3$$

$$w \geq F(\mathbf{y}) = \max_{b, \mathbf{v}} d \cdot b - \sum_{i \in \mathcal{M}} y_i v_i$$

$$b - v_i \leq c_i \quad \forall i \in \mathcal{M}$$

$$b \in \mathbb{R}$$

$$\mathbf{v} \geq 0$$

$$w \geq 0, y_i \in \{0, 1\}$$

# Reformulation Benders 2

$$\begin{aligned} \min \quad & \sum_{i \in \mathcal{M}} f_i y_i + w \\ & y_1 \geq y_2, \quad y_1 \geq y_3 \\ & w \geq F(\mathbf{y}) = \max_{b, \mathbf{v}} d \cdot b - \sum_{i \in \mathcal{M}} y_i v_i \\ & \quad b - v_i \leq c_i \quad \forall i \in \mathcal{M} \\ & \quad b \in \mathbb{R} \\ & \quad \mathbf{v} \geq 0 \\ & w \geq 0, y_i \in \{0, 1\} \end{aligned}$$

Le **max** représente la valeur d'une solution duale optimale  $(b^*, \mathbf{v}^*)$

- L'ensemble de solutions duales réalisables  $(b, \mathbf{v})$  est le même pour tout  $\mathbf{y}$
- Même si  $(b^*, \mathbf{v}^*)$  n'est pas optimale pour tous les  $\mathbf{y}$ , sa valeur objectif reste toujours une borne inférieure pour  $w$

# Reformulation Benders 3

$$\min \sum_{i \in \mathcal{M}} f_i y_i + w$$

$$y_1 \geq y_2, y_1 \geq y_3$$

$$w \geq d \cdot b - \sum_{i \in \mathcal{M}} y_i v_i \quad \underbrace{\forall b \in \mathbb{R}, \mathbf{v} \geq 0 : b - v_i \leq c_i \quad \forall i \in \mathcal{M}}_{\text{Polytope Benders}}$$

$$w \geq 0, y_i \in \{0, 1\}$$

# Reformulation Benders 3

$$\begin{aligned} \min \quad & \sum_{i \in \mathcal{M}} f_i y_i + w \\ & y_1 \geq y_2, y_1 \geq y_3 \\ & w \geq d \cdot b - \sum_{i \in \mathcal{M}} y_i v_i \quad \underbrace{\forall b \in \mathbb{R}, \mathbf{v} \geq 0 : b - v_i \leq c_i \quad \forall i \in \mathcal{M}}_{\text{Polytope Benders}} \\ & w \geq 0, y_i \in \{0, 1\} \end{aligned}$$

D'innombrables solutions du polytope Benders  $\implies$   
d'innombrables contraintes de type  $w \geq \dots$ .

# Reformulation Benders 3

$$\begin{aligned} \min \quad & \sum_{i \in \mathcal{M}} f_i y_i + w \\ & y_1 \geq y_2, y_1 \geq y_3 \\ & w \geq d \cdot b - \sum_{i \in \mathcal{M}} y_i v_i \quad \underbrace{\forall b \in \mathbb{R}, \mathbf{v} \geq 0 : b - v_i \leq c_i \quad \forall i \in \mathcal{M}}_{\text{Polytope Benders}} \\ & w \geq 0, y_i \in \{0, 1\} \end{aligned}$$

D'innombrables solutions du polytope Benders  $\implies$   
d'innombrables contraintes de type  $w \geq \dots$

- 1 Soit  $(\mathbf{y}^*, w^*)$  la solution optimale du maître restreint
- 2 On résout le sous-problème Benders  $F(\mathbf{y}^*)$  et on détermine les valeurs duales optimales  $(b^*, \mathbf{v}^*)$ .
- 3 On ajoute dans le maître la coupe associée à  $(b^*, \mathbf{v}^*)$  :

$$w \geq d \cdot b^* - \sum_i y_i v_i^*$$

- 4 Revenir à 1 si la coupe ci-dessus est violée par  $\mathbf{y}^*$ .



# Questions

1 Si la solution du sous-problème Benders est un rayon de valeur objectif non-bornée ?

- Toutes les solutions avec  $v_i \leq b \forall i$  sont des rayons !

# Questions

1 Si la solution du sous-problème Benders est un rayon de valeur objectif non-bornée ?

- Toutes les solutions avec  $v_i \leq b \forall i$  sont des rayons !

Réponse : on obtient une **coupe de faisabilité**. La plus forte est  $b = v_1 = v_2 \dots = v_n$ , ce qui conduit à  $d - \sum_{i \in \mathcal{M}} y_i \leq 0$ . Cette contrainte impose d'ouvrir au minimum  $d$  sites ; on aurait pu l'ajouter dès le départ.

Les coupes précédentes «  $\dots \geq w$  » étaient des **coupes d'optimalité**.

# Questions

**1** Si la solution du sous-problème Benders est un rayon de valeur objectif non-bornée ?

- Toutes les solutions avec  $v_i \leq b \forall i$  sont des rayons !

Réponse : on obtient une **coupe de faisabilité**. La plus forte est  $b = v_1 = v_2 \dots = v_n$ , ce qui conduit à  $d - \sum_{i \in \mathcal{M}} y_i \leq 0$ . Cette contrainte impose d'ouvrir au minimum  $d$  sites ; on aurait pu l'ajouter dès le départ.

Les coupes précédentes «  $\dots \geq w$  » étaient des **coupes d'optimalité**.

---

**2** Si on a  $y_i = 0$  pour un  $i$ , peut-on avoir une solution du polytope Benders avec  $v_i \rightarrow \infty$  ?

# Questions

**1** Si la solution du sous-problème Benders est un rayon de valeur objectif non-bornée ?

- Toutes les solutions avec  $v_i \leq b \forall i$  sont des rayons !

Réponse : on obtient une **coupe de faisabilité**. La plus forte est  $b = v_1 = v_2 \dots = v_n$ , ce qui conduit à  $d - \sum_{i \in \mathcal{M}} y_i \leq 0$ . Cette contrainte impose d'ouvrir au minimum  $d$  sites ; on aurait pu l'ajouter dès le départ.

Les coupes précédentes «  $\dots \geq w$  » étaient des **coupes d'optimalité**.

---

**2** Si on a  $y_i = 0$  pour un  $i$ , peut-on avoir une solution du polytope Benders avec  $v_i \rightarrow \infty$  ? Oui, il faut faire attention à cela et chercher des coupes avec de petits coefficients.

# Questions

**1** Si la solution du sous-problème Benders est un rayon de valeur objectif non-bornée ?

- Toutes les solutions avec  $v_i \leq b \forall i$  sont des rayons !

Réponse : on obtient une **coupe de faisabilité**. La plus forte est  $b = v_1 = v_2 \dots = v_n$ , ce qui conduit à  $d - \sum_{i \in \mathcal{M}} y_i \leq 0$ . Cette contrainte impose d'ouvrir au minimum  $d$  sites ; on aurait pu l'ajouter dès le départ.

Les coupes précédentes «  $\dots \geq w$  » étaient des **coupes d'optimalité**.

---

**3** Rappel : le coût d'affectation des clients pour un  $\mathbf{y}^*$  figé est  $F(\mathbf{y}^*) = d \cdot b^* - \sum_i y_i^* v_i^*$ . Donc la dérivée partielle de  $F(\mathbf{y})$  par rapport à tout  $y_i$  est  $\leq 0$ . C'est normal ?

# Questions

**1** Si la solution du sous-problème Benders est un rayon de valeur objectif non-bornée ?

- Toutes les solutions avec  $v_i \leq b \forall i$  sont des rayons !

Réponse : on obtient une **coupe de faisabilité**. La plus forte est  $b = v_1 = v_2 \dots = v_n$ , ce qui conduit à  $d - \sum_{i \in \mathcal{M}} y_i \leq 0$ . Cette contrainte impose d'ouvrir au minimum  $d$  sites ; on aurait pu l'ajouter dès le départ.

Les coupes précédentes «  $\dots \geq w$  » étaient des **coupes d'optimalité**.

---

**3** Rappel : le coût d'affectation des clients pour un  $\mathbf{y}^*$  figé est  $F(\mathbf{y}^*) = d \cdot b^* - \sum_i y_i^* v_i^*$ . Donc la dérivée partielle de  $F(\mathbf{y})$  par rapport à tout  $y_i$  est  $\leq 0$ . C'est normal ? Oui : plus de sites  $\implies$  le coût d'affectation ne peut que baisser

# Benders automatique

`cplex ≥ 12.9` est capable de réaliser une décomposition de Benders automatique. Si vous déclarez les variables **y** comme des variables entières et les variables **x** comme des variables continues, il va jouer sur les variables **x** pour générer très probablement la même décomposition que nous.

Sous C++, il suffit d'ajouter :

```
cplex.setParam( IloCplex::Param::Benders::  
    Strategy , IloCplex::BendersFull );
```

# Résoudre le sous-problème plus vite sans PL

si  $\sum_i y_i = d$  ?

Rappel :

$$\begin{aligned} F(\mathbf{y}) = \max_{b, \mathbf{v}} & d \cdot b - \sum_{i \in \mathcal{M}} y_i v_i \\ & b - v_i \leq c_i \quad \forall i \in \mathcal{M} \\ & b \in \mathbb{R} \\ & \mathbf{v} \geq 0 \end{aligned}$$

Soit la solution suivante  $b = \max_i c_i$  et  $v_i = b - c_i \forall i \in [1..n]$  :

Sa valeur objectif est  $\sum y_i c_i$ , c.a.d, on ne peut pas faire mieux quelque soit  $\mathbf{y}$  avec  $\sum_i y_i = d$  !

Une seule contrainte peut suffire !



# Résoudre le sous-problème plus vite sans PL

si  $\sum_i y_i = d$  ?

Rappel :

$$\begin{aligned} F(\mathbf{y}) = \max_{b, \mathbf{v}} & d \cdot b - \sum_{i \in \mathcal{M}} y_i v_i \\ & b - v_i \leq c_i \quad \forall i \in \mathcal{M} \\ & b \in \mathbb{R} \\ & \mathbf{v} \geq 0 \end{aligned}$$

Soit la solution suivante  $b = \max_i c_i$  et  $v_i = b - c_i \forall i \in [1..n]$  :  
Sa valeur objectif est  $\sum y_i c_i$ , c.a.d, on ne peut pas faire mieux  
quelque soit  $\mathbf{y}$  avec  $\sum_i y_i = d$  !

Une seule contrainte peut suffire !

# Résoudre le sous-problème plus vite sans PL

si  $\sum_i y_i = d$ ?

Rappel :

$$\begin{aligned} F(\mathbf{y}) = \max_{b, \mathbf{v}} & d \cdot b - \sum_{i \in \mathcal{M}} y_i v_i \\ & b - v_i \leq c_i \quad \forall i \in \mathcal{M} \\ & b \in \mathbb{R} \\ & \mathbf{v} \geq 0 \end{aligned}$$

Soit la solution suivante  $b = \max_i c_i$  et  $v_i = b - c_i \forall i \in [1..n]$  :  
Sa valeur objectif est  $\sum y_i c_i$ , c.a.d, on ne peut pas faire mieux  
quelque soit  $\mathbf{y}$  avec  $\sum_i y_i = d$ !

Une seule contrainte peut suffire !

# Comparaison sur ma machine

Pour  $n = 50.000$

0.8s Modèle de base sans décomposition `cplex 12.6`

10s Modèle de base sans décomposition `cplex 12.10`

---

1.6s Décomposition automatique `cplex 12.10`

---

0.3s Décomposition Benders manuelle résolution sous-problème sans PL `cplex 12.6`

0.3s Décomposition Benders manuelle résolution sous-problème sans PL `gurobi8.11`

>10s Décomposition Benders manuelle résolution sous-problème par PL `cplex 12.6`

# Sur la vitesse du code

La théorie seule ne peut pas tout expliquer :

- Le temps de calcul est multiplié par 7 si on oublie d'appeler `cplex.end()` à la fin du sous-problème.
  - **Video :** [youtu.be/YuKnE6zH-J8](https://youtu.be/YuKnE6zH-J8)
- Lorsqu'on résout le sous-problème, il faut chercher une coupe avec de petits coefficients. Après avoir optimisé l'objectif de départ `obj`, on ajoute `model.add(obj==objVal)` et on change la fonction objectif. Deux approches pour cela
  - 1 un seul appel `ilo_obj.setLinearCoefs(v,newvals)`
  - 2 une boucle qui appelle `ilo_obj.setLinearCoef(...)` sur chaque variable

La version 2 ne s'exécute pas en temps constant mais en temps linéaire ; ainsi, un « simple » changement de fonction objectif peut nécessiter un temps quadratique.

# Sur la vitesse du code

La théorie seule ne peut pas tout expliquer :

- Le temps de calcul est multiplié par 7 si on oublie d'appeler `cplex.end()` à la fin du sous-problème.
  - Video : [youtu.be/YuKnE6zH-J8](https://youtu.be/YuKnE6zH-J8)
- Lorsqu'on résout le sous-problème, il faut chercher une coupe avec de petits coefficients. Après avoir optimisé l'objectif de départ `obj`, on ajoute `model.add(obj==objVal)` et on change la fonction objectif. Deux approches pour cela
  - 1 un seul appel `ilo_obj.setLinearCoefs(v,newvals)`
  - 2 une boucle qui appelle `ilo_obj.setLinearCoef(...)` sur chaque variable

La version 2 ne s'exécute pas en temps constant mais en temps linéaire ; ainsi, un « simple » changement de fonction objectif peut nécessiter un temps quadratique.

# Sur la vitesse du code

La théorie seule ne peut pas tout expliquer :

- Le temps de calcul est multiplié par 7 si on oublie d'appeler `cplex.end()` à la fin du sous-problème.
  - Video : [youtu.be/YuKnE6zH-J8](https://youtu.be/YuKnE6zH-J8)
- Lorsqu'on résout le sous-problème, il faut chercher une coupe avec de petits coefficients. Après avoir optimisé l'objectif de départ `obj`, on ajoute `model.add(obj==objVal)` et on change la fonction objectif. Deux approches pour cela
  - 1 un seul appel `ilo_obj.setLinearCoefs(v,newvals)`
  - 2 une boucle qui appelle `ilo_obj.setLinearCoef(...)` sur chaque variable

La version 2 ne s'exécute pas en temps constant mais en temps linéaire ; ainsi, un « simple » changement de fonction objectif peut nécessiter un temps quadratique.