

Optimisation

ESP (Ecole Supérieure Polytechnique) de Dakar

Alain Faye

4 – Programmation linéaire en nombres entiers

Contenu

- I. Procédures arborescentes : principes généraux
- II. Programmation Linéaire en Nombres Entiers
 - 1. Résolution par procédure arborescente
 - 2. Algorithme de coupes

Procédures arborescentes

Problèmes en variables discrètes

Enumération implicite

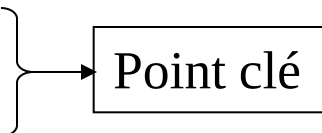
Enumération totale évitée à l'aide de bornes inférieures et bornes supérieures

Problème P : minimiser z

UB = valeur de z pour une solution de P

LB borne inférieure du minimum de z

si $LB \geq UB$ alors on ne peut avoir mieux que UB
et UB est la valeur de la solution optimale



On énumère ce qui revient à diviser P en sous-problèmes de plus en plus petits

Quand un problème est petit on trouve facilement sa solution optimale UB

On garde le meilleur UB trouvé

Pour chacun des sous-problèmes on calcule LB et on applique le test $LB \geq UB$

C'est ce test qui permet d'éviter l'énumération totale en écartant les sous-problèmes pas intéressants

Exemple

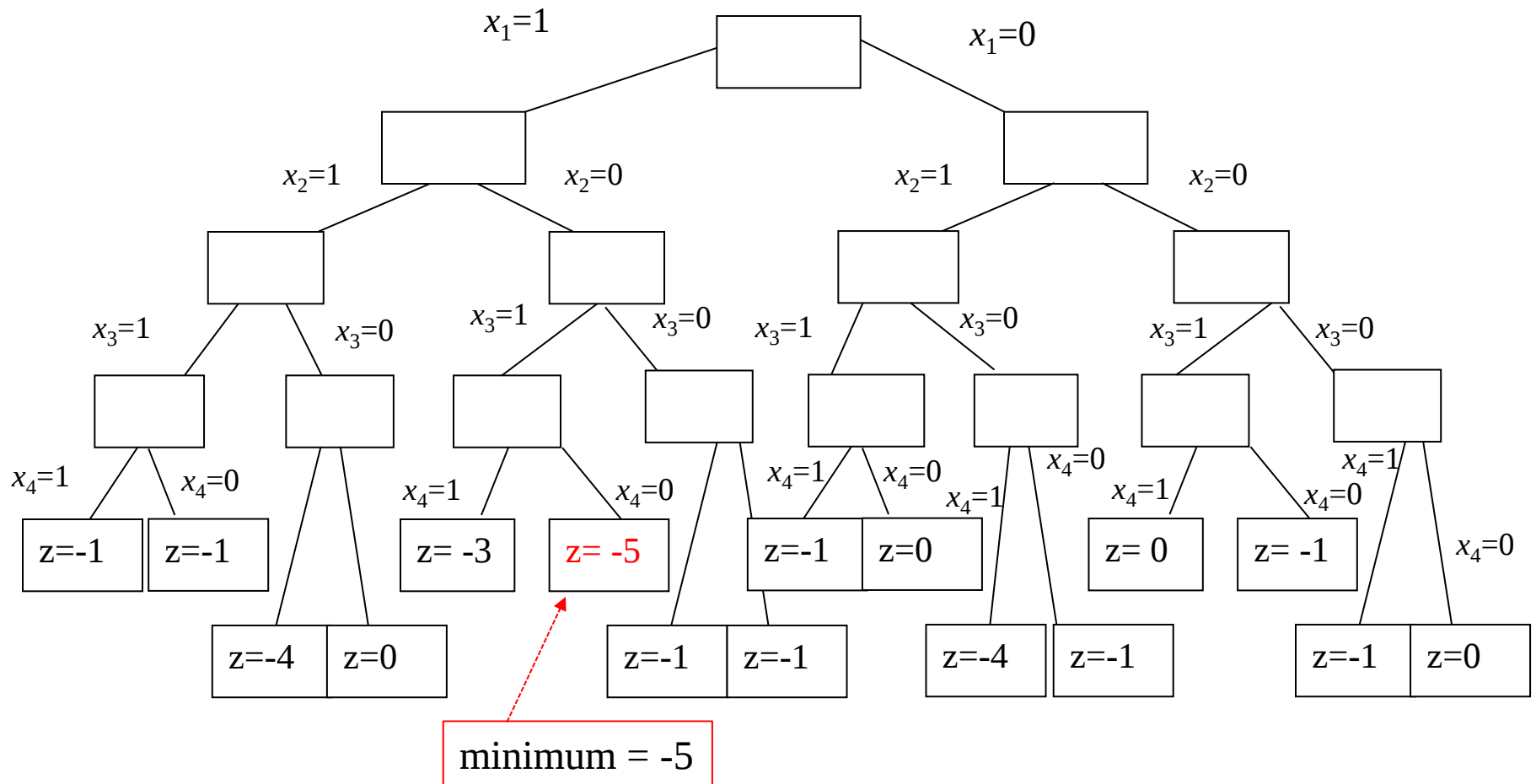
$\min z = -x_1 - x_2 - x_3 - x_4 + 3x_1x_2 - 3x_1x_3 + x_1x_4 + 2x_2x_3 - 2x_2x_4 + 2x_3x_4$ avec $x_1, x_2, x_3, x_4 \in \{0, 1\}$

Enumération

- on sépare sur une variable x_i que l'on fixe à 1 ou à 0.

On prend les variables dans l'ordre x_1, x_2, x_3, x_4 .

Enumération totale



Peut-on éviter de faire les 2⁴ cas ?

Exemple

$\min z = -x_1 - x_2 - x_3 - x_4 + 3x_1x_2 - 3x_1x_3 + x_1x_4 + 2x_2x_3 - 2x_2x_4 + 2x_3x_4$ avec $x_1, x_2, x_3, x_4 \in \{0, 1\}$

- **LB** de z = la valeur constituée par l'ensemble des variables fixées
+ la somme des coefficients < 0 relatifs aux variables libres restantes.

À la racine (aucune variable fixée), $LB = -1 - 1 - 1 - 1 - 3 - 2 = -9$

pour $x_1=1, x_2=0$ et x_3, x_4 libres, $z = -1 - x_3 - x_4 - 3x_3 + x_4 + 2x_3x_4$,

$z = -1 - x_3 - 3x_3 + 2x_3x_4 \geq -5$ et $LB = -5$,

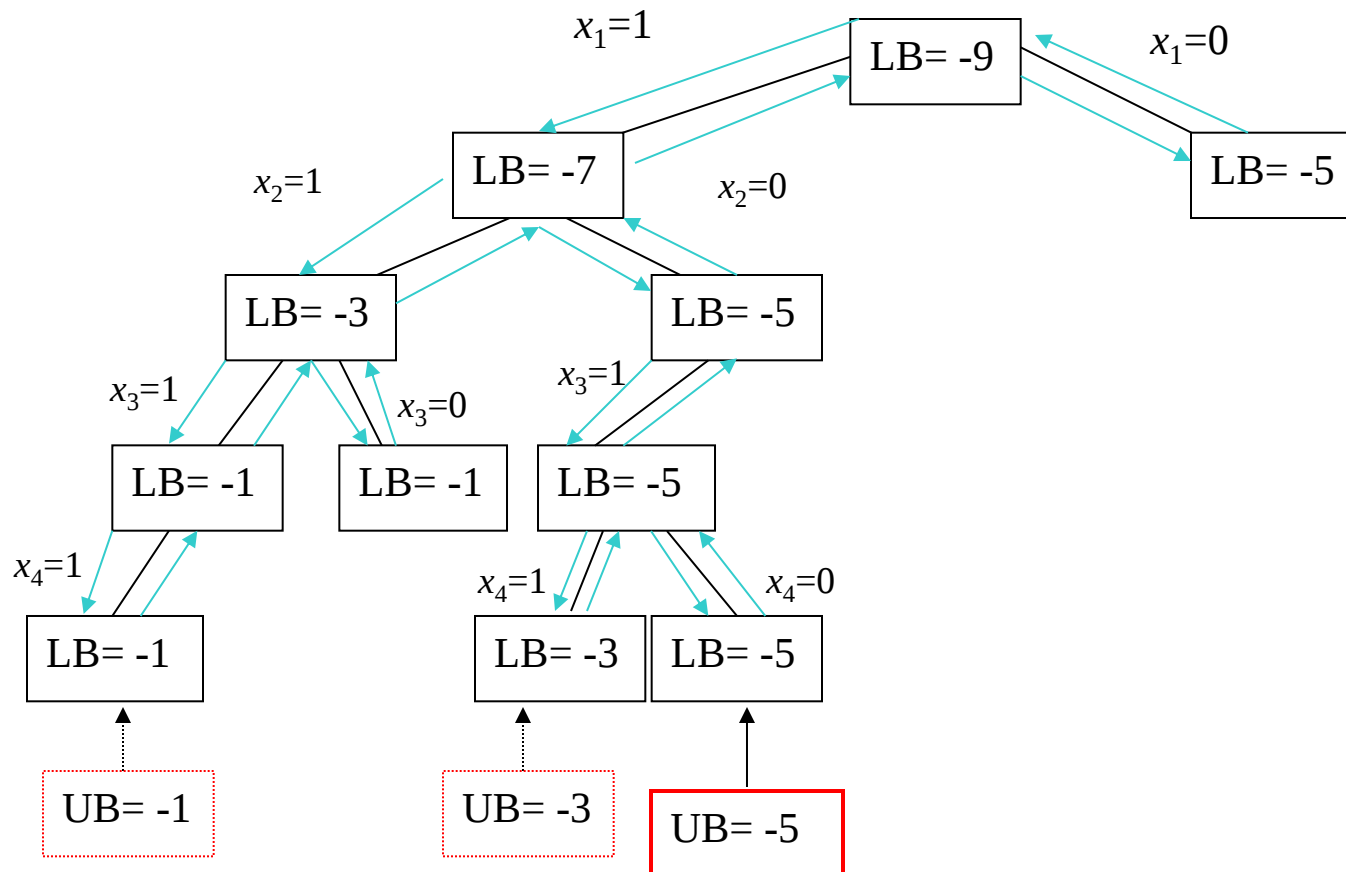
pour $x_1=1, x_2=1$ et x_3, x_4 libres, $z = -1 - 1 - x_3 - x_4 + 3 - 3x_3 + x_4 + 2x_3 - 2x_4 + 2x_3x_4$

$z = 1 - 2x_3 - 2x_4 + 2x_3x_4 \geq -3$ et $LB = -3$.

LB se calcule facilement (algorithme de complexité linéaire)

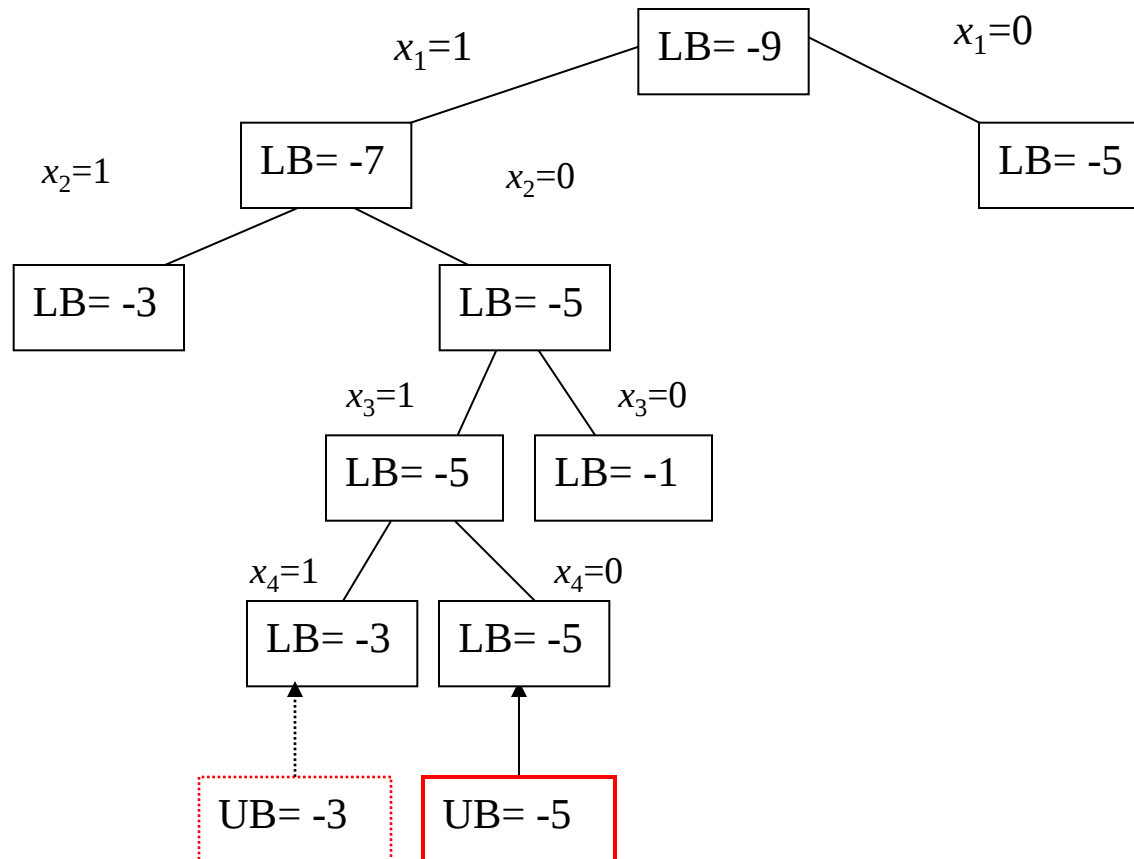
- **UB** est calculé à chaque fois que l'on est sur une feuille de l'arbre de recherche.

Recherche en profondeur d 'abord



On rencontre $UB = -1$, -3 et finalement -5 et on prouve qu 'il n 'y a pas mieux

Recherche en meilleur d 'abord



On rencontre $\text{UB} = -3$ et finalement -5 et on prouve qu 'il n 'y a pas mieux

Procédures arborescentes en minimisation - résumé

- **Les bornes**

LB facile à calculer, calculée en chaque nœud de l'arbre

UB obtenue aux feuilles de l'arbre ou calculée par un algo simple

- **Séparation**

On énumère, on divise le pb ou sous-pb de plus en plus petits

- **Elagage**

de l'arbre de recherche grâce aux bornes UB et LB

Si $LB \geq UB$ en un nœud de l'arbre on coupe ce nœud.

Programmation linéaire en nombres entiers

$$(P) \quad \min/\max z=cx \text{ s.c. } Ax \leq b \quad x \in \mathbb{N}^n$$

c un vecteur ligne de n coordonnées,

A une matrice m lignes, n colonnes,

b un vecteur colonne de m coordonnées,

x un vecteur colonne de n coordonnées représentant les variables du problème.

$\mathbb{N}$ désigne ici l'ensemble des entiers naturels $\{0,1,2,3 \dots\}$.

Les coefficients de c , A , b sont supposés entiers (pas forcément positifs).

Résolution:

- procédure arborescente
- algorithme des coupes de Gomory

Programmation linéaire en nombres entiers – Les applications

Beaucoup de problèmes se modélisent par la PLNE

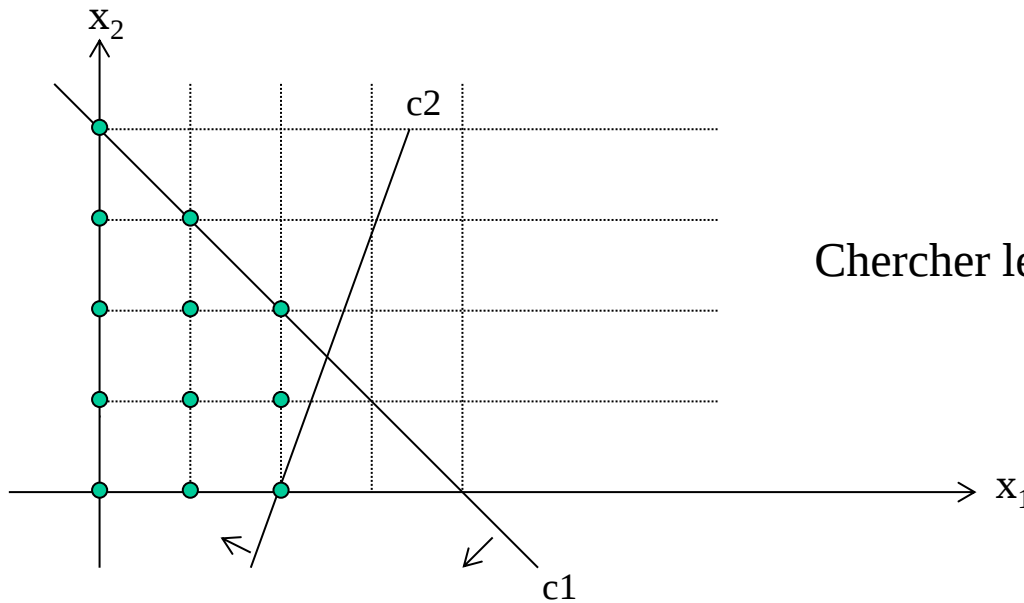
Exemples:

- Localisation usines, entrepôts, magasins
 - Localisation de matériels dans les réseaux de télécoms
 - Ordonnancements de tâches , allocation de ressources
 - Emploi du temps
 - Contrôle aérien, séquençement des avions
- ...

PLNE - Exemple

$$\min z = -2x_1 - x_2$$

$$\text{s.c.} \begin{cases} x_1 + x_2 \leq 4 & (\text{c1}) \\ 3x_1 - x_2 \leq 6 & (\text{c2}) \\ x_1 \in \mathbb{N} \quad x_2 \in \mathbb{N} \end{cases}$$



Chercher le point vert qui minimise z

Procédure arborescente

Pb de type minimiser

Évaluation LB

LB = valeur de la relaxation continue du problème où $x \in \mathbb{N}$ est relaxée en $x \geq 0$, on a un PL que l'on résout par l'algorithme du simplexe (par exemple)

Séparation

si une variable x_i est fractionnaire on crée 2 sous-problèmes

par exemple: $x_i = 7/4$

on crée un sous-problème avec la contrainte $x_i \leq \lfloor 7/4 \rfloor = 1$

l'autre avec la contrainte $x_i \geq \lfloor 7/4 \rfloor + 1 = 2$

Majorant UB

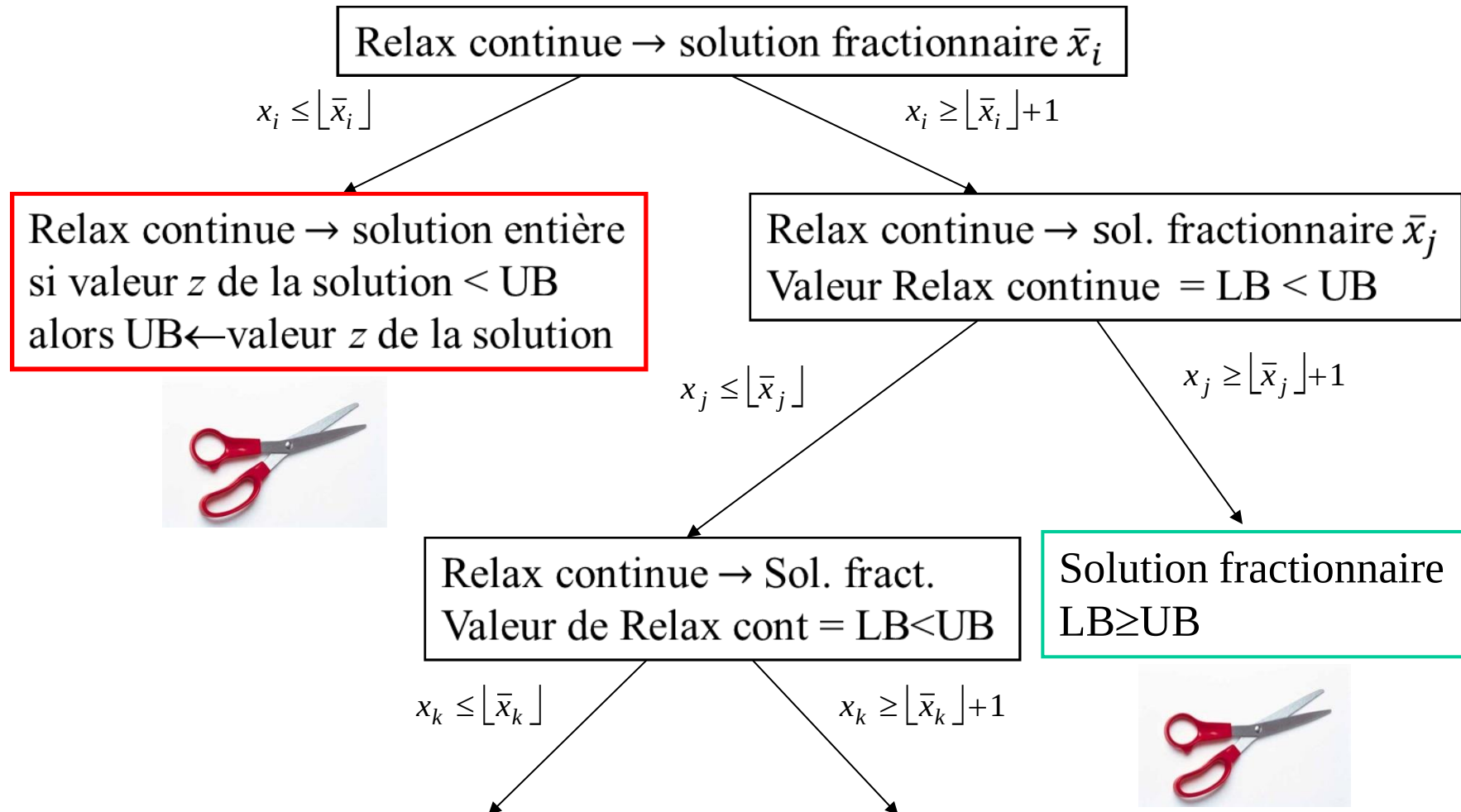
Lorsque la résolution d'un sous-problème relaxé donne une solution entière, on obtient une solution.

Si elle est meilleure que la précédente on la mémorise (UB)

Elagage de l'arbre de recherche

Quand un sous-problème donne une évaluation $LB \geq UB$ on l'élimine.

Problème de minimisation – Résumé des cas possibles

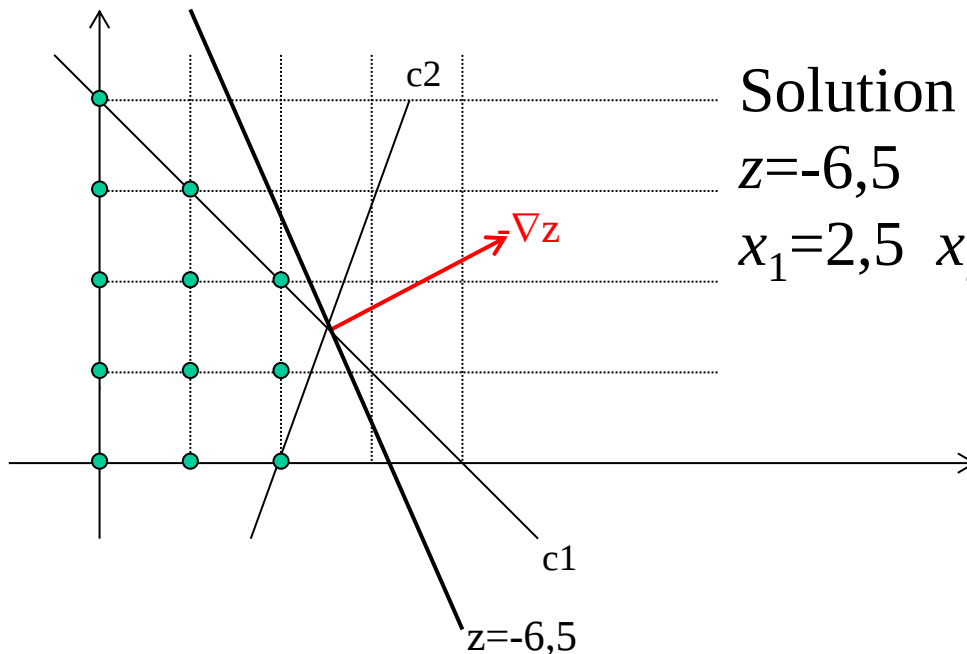


PLNE - Exemple

Relaxation continue $x_1 \geq 0, x_2 \geq 0$

$$\min z = -2x_1 - x_2$$

$$\text{s.c.} \begin{cases} x_1 + x_2 \leq 4 & (\text{c1}) \\ 3x_1 - x_2 \leq 6 & (\text{c2}) \\ x_1 \geq 0 \quad x_2 \geq 0 \end{cases}$$



Solution optimale relax continue

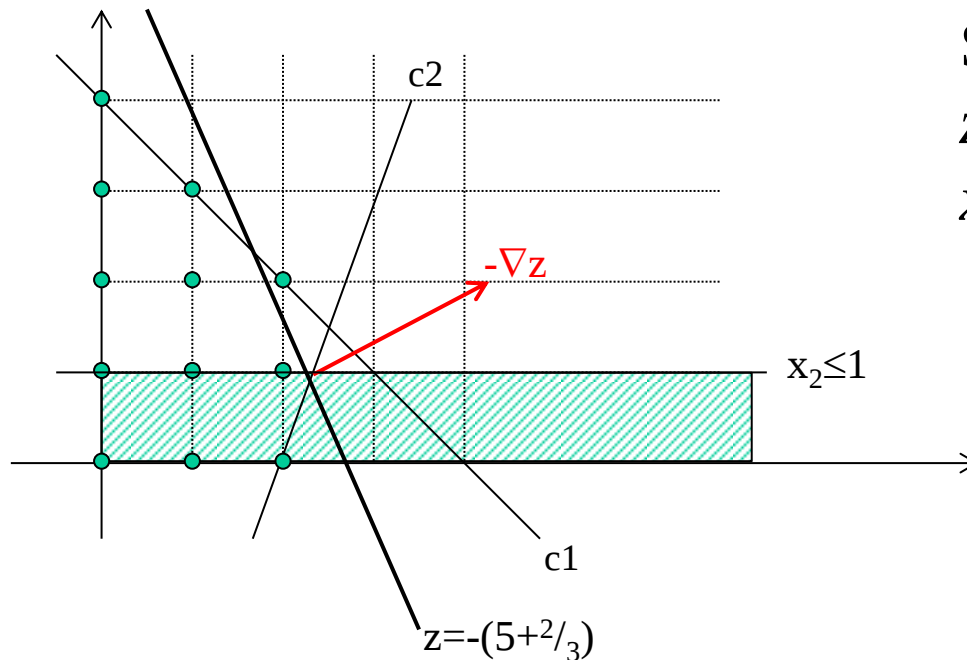
$$z = -6,5$$

$$x_1 = 2,5 \quad x_2 = 1,5$$

On choisit une variable fractionnaire par exemple $x_2=1,5$

On crée deux sous-problèmes

$x_2 \leq 1$ et $x_2 \geq 2$



Sous-problème $x_2 \leq 1$

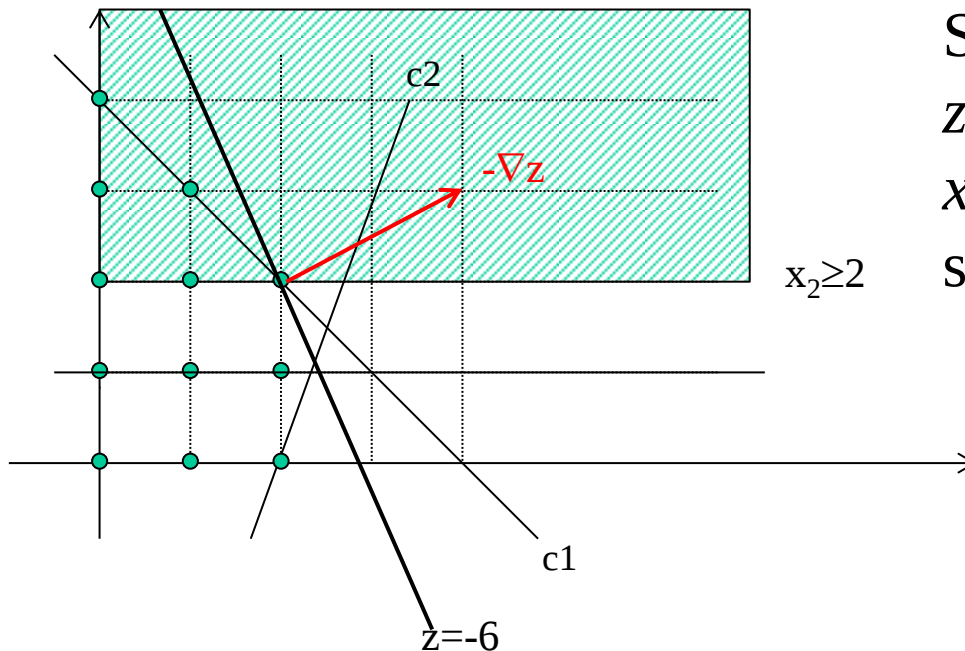
$z = -(5 + 2/3)$

$x_1 = 2 + 1/3$ $x_2 = 1$

On choisit une variable fractionnaire par exemple $x_2=1,5$

On crée deux sous-problèmes

$x_2 \leq 1$ et $x_2 \geq 2$



Sous-problème $x_2 \geq 2$

$z = -6$

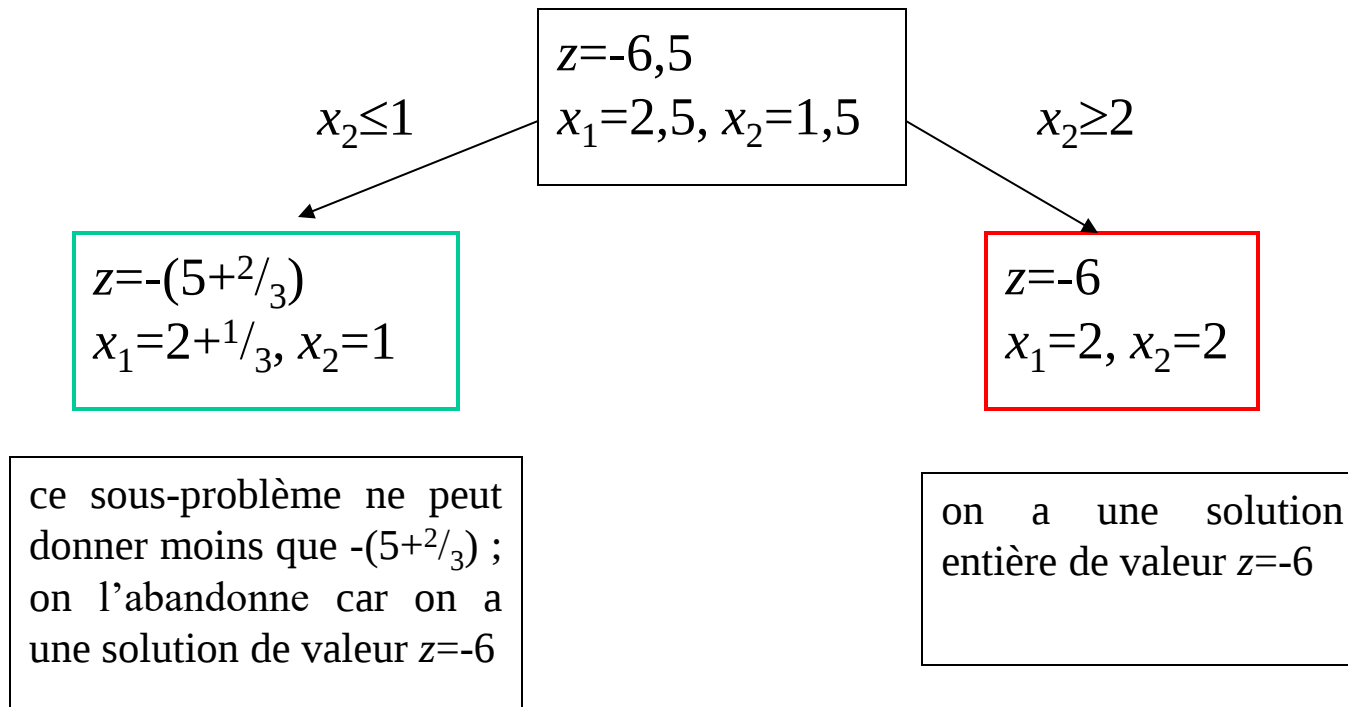
$x_1 = 2$ $x_2 = 2$

solution entière

PLNE – Exemple - bilan

$$\min z = -2x_1 - x_2$$

$$\text{s.c.} \begin{cases} x_1 + x_2 \leq 4 & (\text{c 1}) \\ 3x_1 - x_2 \leq 6 & (\text{c 2}) \\ x_1 \in \mathbb{N} \quad x_2 \in \mathbb{N} \end{cases}$$



Inégalités valides

$$(P) \quad \min/\max z=cx \text{ s.c. } Ax \leq b \quad x \in \mathbb{N}^n$$

On note $F(P)$ l'ensemble des solutions réalisables du pb P

$F(P_R)$ l'ensemble des solutions réalisables de la relax continue de P

Inégalité valide $\pi x \leq \pi_0$ si elle est vérifiée $\forall x \in F(P)$

Inégalité valide intéressante si elle « tronque » $F(P_R)$

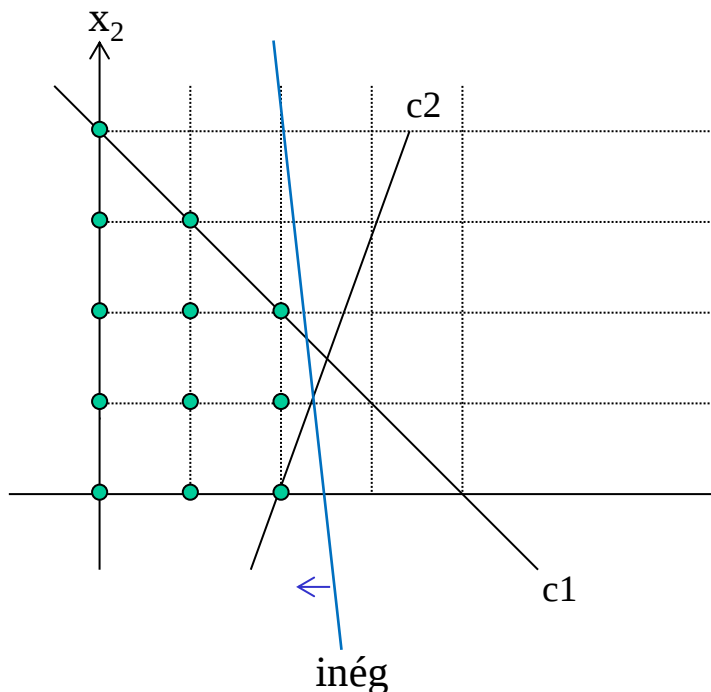
L'ajout d'inégalité valides améliore la relax continue

LB plus haute donc on tronque l'arbre de recherche plus facilement

Inégalités valides - exemple

$$\min z = -2x_1 - x_2$$

$$\text{s.c.} \begin{cases} x_1 + x_2 \leq 4 & (c1) \\ 3x_1 - x_2 \leq 6 & (c2) \\ x_1 \in \mathbb{N} \quad x_2 \in \mathbb{N} \end{cases}$$



$F(P)$ = les points verts

Tous les points verts vérifient inég

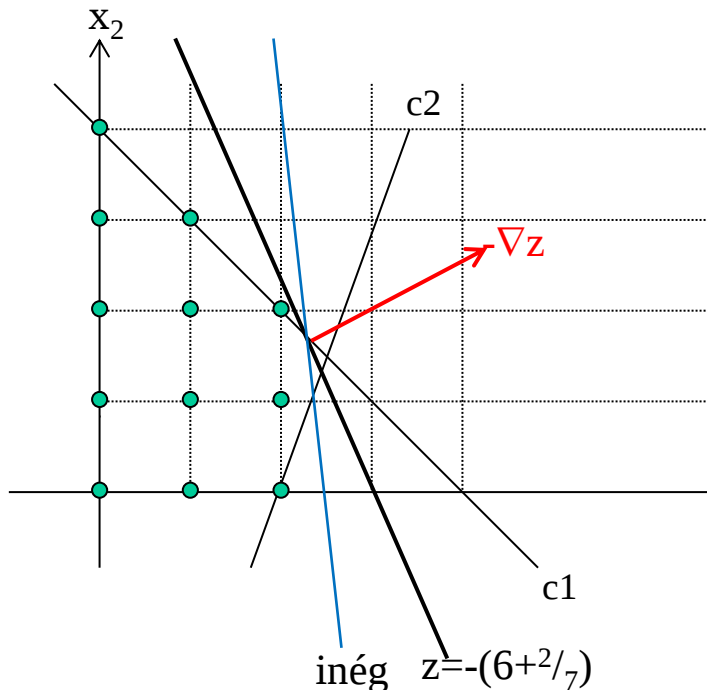
Des points de $F(P_R)$ ne satisfont pas inég
par exemple $x_1=2,5 \quad x_2=1,5$

Inégalité valide $8x_1 + x_2 \leq 20$ (inég)

Inégalité valide améliore relax continue - exemple

$$\min z = -2x_1 - x_2$$

$$\text{s.c.} \begin{cases} x_1 + x_2 \leq 4 & (\text{c1}) \\ 3x_1 - x_2 \leq 6 & (\text{c2}) \\ 2x_1 + x_2 \leq 6 & (\text{inég}) \\ x_1 \geq 0, x_2 \geq 0 \end{cases}$$



Relax continue + inég

$$z = -(6 + \frac{2}{7})$$

$$x_1 = \frac{16}{7} \quad x_2 = \frac{12}{7}$$

Inégalité valide - coupe de Chvatal

Notation: $\lfloor \alpha \rfloor$ = plus petit entier inférieur ou égal à α

exemples: $\lfloor 5,4 \rfloor = 5$ $\lfloor -5,4 \rfloor = -6$ $\lfloor 5 \rfloor = 5$

Coupes de Chvatal donnent inégalités valides pour $F(P)$ à partir d'inégalités valides pour $F(P_R)$ de la façon suivante:

Soit $\sum_j a_j x_j \leq \beta$ une inégalité valide pour $F(P_R)$

- **relaxation membre gauche**

$\sum_j \lfloor a_j \rfloor x_j \leq \beta$ est valide car $x \geq 0$

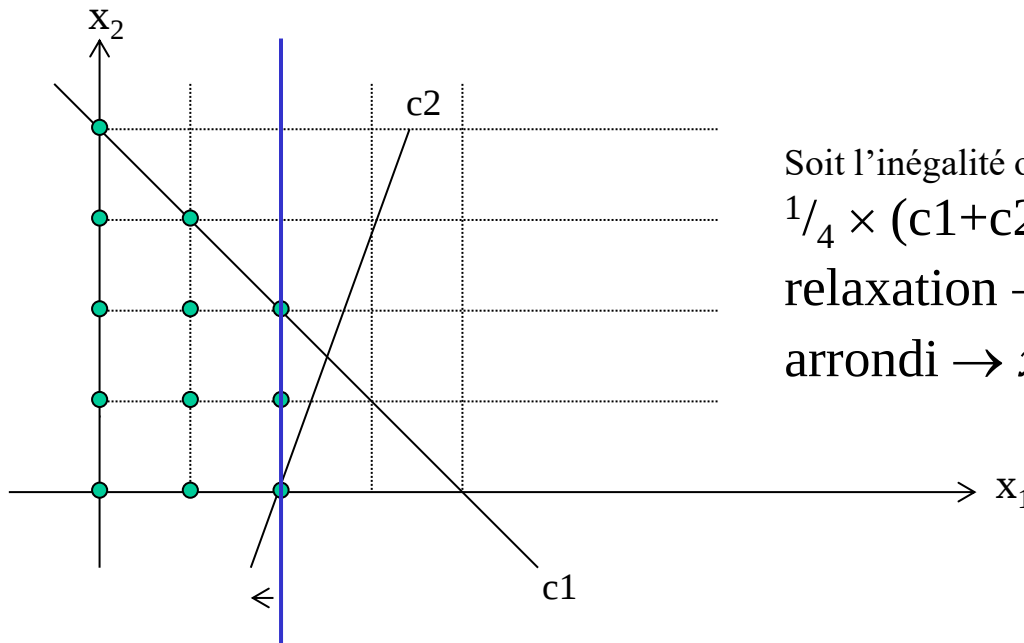
- **arrondi membre droit** (c'est ici que l'on coupe)

$\sum_j \lfloor a_j \rfloor x_j \leq \lfloor \beta \rfloor$ est valide pour $F(P)$ car x entier

Inégalité valide - coupe de Chvatal - exemple

$$\min z = -2x_1 - x_2$$

$$\text{s.c.} \begin{cases} x_1 + x_2 \leq 4 & (c1) \\ 3x_1 - x_2 \leq 6 & (c2) \\ x_1 \in \mathbb{N} \quad x_2 \in \mathbb{N} \end{cases}$$



Soit l'inégalité obtenue par combinaison de c1 et c2

$$\frac{1}{4} \times (c1 + c2) \rightarrow \frac{4}{4}x_1 \leq \frac{10}{4}$$

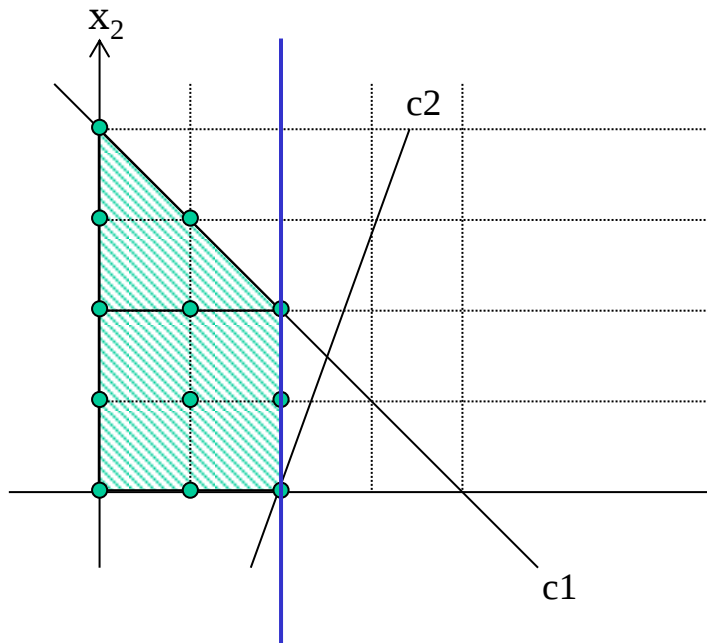
$$\text{relaxation} \rightarrow x_1 \leq \frac{10}{4}$$

$$\text{arrondi} \rightarrow x_1 \leq 2$$

Coupe de Chvatal $x_1 \leq 2$

Inégalité valide - coupe de Chvatal

En rajoutant des coupes de Chvatal,
on arrive à l'enveloppe convexe de $F(P)$
en un nombre fini d'itérations (ajout d'un nombre fini de coupes)



Coupe de Chvatal $x_1 \leq 2$

$$\text{Conv}(F(P)) = \begin{cases} x_1 + x_2 \leq 4 & (c1) \\ 3x_1 - x_2 \leq 6 & (c2) \\ x_1 \leq 2 & (\text{Chvatal}) \\ x_1 \geq 0, x_2 \geq 0 \end{cases}$$

Algorithme des coupes de Gomory

1. On résout P_R la relaxation continue de P

Si la solution x_R^* est entière alors P est résolu et on s'arrête

Sinon

- on ajoute une coupe de Chvatal
qui élimine cette solution x_R^* fractionnaire
- on retourne en 1

Algorithme des coupes de Gomory - exemple

$$\begin{aligned} \min z &= -2x_1 - x_2 \\ \text{s.c. } \left\{ \begin{array}{lll} x_1 & + x_2 & \leq 4 \quad (\text{c 1}) \\ 3x_1 & - x_2 & \leq 6 \quad (\text{c 2}) \\ x_1 \in \mathbb{N} & x_2 \in \mathbb{N} & \end{array} \right. \end{aligned}$$

Mise du pb sous forme standard (contraintes d'égalités)

$$\begin{aligned} \min z &= -2x_1 - x_2 \\ \text{s.c. } \left\{ \begin{array}{llll} x_1 & + x_2 & + x_3 & = 4 \\ 3x_1 & - x_2 & & + x_4 = 6 \\ x_1 \in \mathbb{N} & x_2 \in \mathbb{N} & x_3 \in \mathbb{N} & x_4 \in \mathbb{N} \end{array} \right. \end{aligned}$$

Solution de la relaxation continue $x_1=5/2$, $x_2=3/2$, $x_3=x_4=0$

base	x_1	x_2	x_3	x_4	
x_1	1		$\frac{1}{4}$	$\frac{1}{4}$	$= \frac{5}{2}$
x_2		1	$\frac{3}{4}$	$-\frac{1}{4}$	$= \frac{3}{2}$
			$\frac{5}{4}$	$\frac{1}{4}$	$= \frac{13}{2} + z$

fractionnaire

$$x_1 + \frac{1}{4}x_3 + \frac{1}{4}x_4 = \frac{5}{2}$$

$$x_1 + (0 + \frac{1}{4})x_3 + (0 + \frac{1}{4})x_4 = 2 + \frac{1}{2}$$

$$x_1 + (0)x_3 + (0)x_4 \leq 2 + \frac{1}{2}$$

$$x_1 + (0)x_3 + (0)x_4 \leq 2$$

(on coupe ici)

+

-

$$\frac{1}{4}x_3 + \frac{1}{4}x_4 \geq \frac{1}{2}$$

La solution de base ne vérifie pas cette inégalité
Elle sera exclue par cette inégalité

Branch-and-Cut

On combine procédure arborescente et ajout de coupes

Procédure arborescente

+ ajout de coupes en chaque nœud de l'arborescence

Ajout de coupes permet d'améliorer LB en chaque nœud

Si on augmente LB on élague plus l'arbre de recherche

En pratique, on met un nombre limité de coupes en chaque nœud pour économiser le temps de calcul

Souvent on ne met des coupes qu'à la racine de l'arborescence

Algorithme à 2 phases : coupes uniquement à la racine

1. Génération de coupes . On obtient un polyèdre P.
2. Si on a obtenu une solution entière alors STOP
Sinon faire du Branch&Bound à partir de P pour trouver une solution entière

Algorithme Branch&Cut

C'est un B&B mais à chaque nœud de l'arborescence de recherche on génère des coupes
Ceci dans le but d'améliorer la valeur de la relaxation continue et d'élaguer plus l'arborescence de recherche.

Exemple: sac-à-dos 0-1

$$\begin{aligned} \max & 7x_1 + 6x_2 + 5x_3 + 4x_4 + 3x_5 + 2x_6 + x_7 \\ \text{s.c. } & x_1 + 2x_2 + 3x_3 + 4x_4 + 5x_5 + 6x_6 + 7x_7 \leq 11 \\ & x \in \{0,1\}^7 \end{aligned}$$

Algorithme à 2 phases

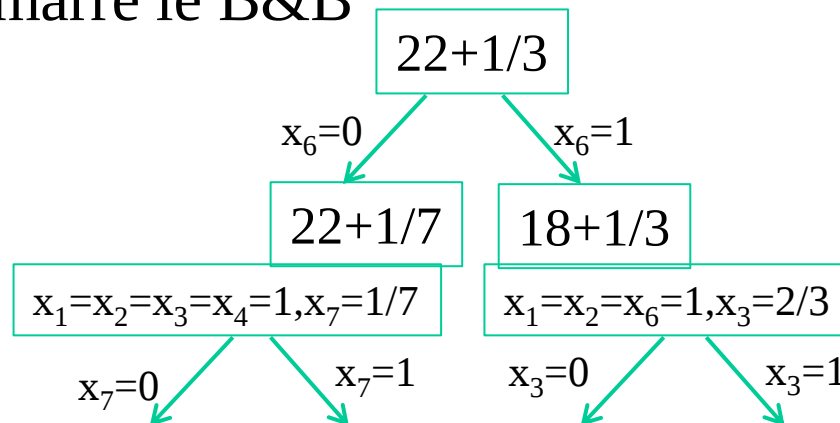
Relaxation continue = 22.6 $x_1=x_2=x_3=x_4=1, x_5=0.2$

On trouve une inégalité de couverture $x_1+x_2+x_3+x_4+x_5 \leq 4$

On la rajoute au modèle

Relaxation continue = $22 + 1/3$ $x_1=x_2=x_3=x_4=1, x_6=1/6$

On démarre le B&B

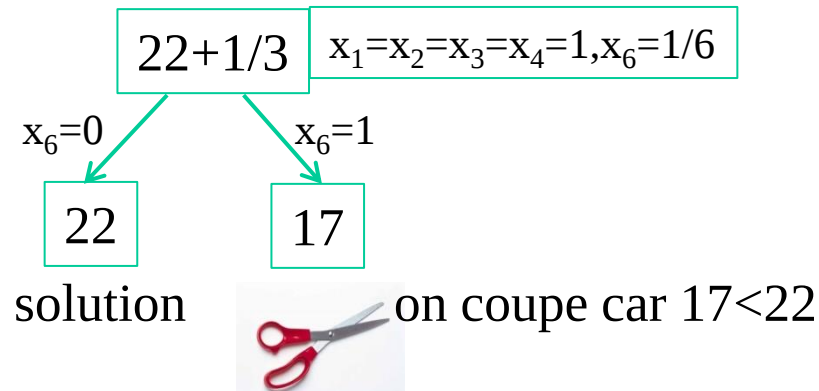


Exemple: sac-à-dos 0-1

$$\begin{aligned} \max & 7x_1 + 6x_2 + 5x_3 + 4x_4 + 3x_5 + 2x_6 + x_7 \\ \text{s.c. } & x_1 + 2x_2 + 3x_3 + 4x_4 + 5x_5 + 6x_6 + 7x_7 \leq 11 \\ & x \in \{0,1\}^7 \end{aligned}$$

Algorithme Branch&Cut

- Le sous-pb $x_6=0$, on trouve une inégalité de couverture $x_1+x_2+x_3+x_7 \leq 3$
Relaxation continue=22 $x_1=x_2=x_3=x_4=1 \Rightarrow$ solution entière
- Le sous-pb $x_6=1$, on trouve une inégalité de couverture $x_1+x_2+x_3 \leq 2$
Relaxation continue=17 $x_1=x_2=x_6=1, x_4=1/2$



Bilan sur la PL et la PLNE

PL : $\min cx \text{ s.c. } Ax \geq b \ x \geq 0$

PLNE : PL + contraintes d'intégrité $x \in \{0, 1, 2, \dots\}$

Problèmes de nature différente:

PL $\rightarrow$ continu

PLNE $\rightarrow$ discret

Algorithmes

- PL :** algorithme du simplexe
- on passe de points extrêmes en points extrêmes voisins
 - cheminement sur la frontière du polyèdre

autres algorithmes : points intérieurs

- cheminement par l 'intérieur du polyèdre

PLNE : procédures arborescentes
 méthodes de coupes
 mixage des 2 approches

Applications

PL : planification , production , transport

PLNE : items indivisibles (non fractionnaires)

problèmes de décision (variables 0-1)

Dualité

Dualité :

relaxation des contraintes + injection dans l 'objectif

en **PL** → pas de saut de dualité

en **PLNE** → existence de saut de dualité

Intérêt :

var. duales mesurent l 'impact des contraintes sur l 'optimum

évaluation de l 'optimum (minorant pour un pb. minimiser)

utilisation dans procédures arborescentes

Solveurs-Logiciels de résolution PL et PLNE

Résolution de problèmes de grandes tailles :
nombre élevé de variables et de contraintes

Langages de modélisation : MathProg, AMPL, Mosel
écriture au format « mathématique » du problème

Solveurs:

XPRESS-MP : Sociétés FICO - Artelys

CPLEX : société IBM

Versions « étudiantes » gratuites

Logiciels libres

COIN-OR

GLPK

Annexes

- Déroulement de l'algorithme des coupes de Gomory sur un exemple
- Un exemple de modélisation PLNE – localisation d'un magasin

Coupe de Gomory

$$\max z = 2x_1 + x_2$$

$$\text{s.c.} \begin{cases} x_1 + x_2 \leq 4 & (\text{contrainte 1}) \\ 3x_1 - x_2 \leq 6 & (\text{contrainte 2}) \\ x_1 \in \mathbb{N} \quad x_2 \in \mathbb{N} \end{cases}$$

$$\max z = 2x_1 + x_2$$

$$\text{s.c.} \begin{cases} x_1 + x_2 + x_3 = 4 \\ 3x_1 - x_2 + x_4 = 6 \\ x_1 \in \mathbb{N} \quad x_2 \in \mathbb{N} \quad x_3 \in \mathbb{N} \quad x_4 \in \mathbb{N} \end{cases}$$

Relaxation continue

base	x_1	x_2	x_3	x_4	
x_1	1		$\frac{1}{4}$	$\frac{1}{4}$	$\frac{5}{2}$
x_2		1	$\frac{3}{4}$	$-\frac{1}{4}$	$\frac{3}{2}$
			$-\frac{5}{4}$	$-\frac{1}{4}$	$-\frac{13}{2} + z$

fractionnaire

$$x_1 + \frac{1}{4}x_3 + \frac{1}{4}x_4 = \frac{5}{2}$$

$$x_1 + (0 + \frac{1}{4})x_3 + (0 + \frac{1}{4})x_4 = 2 + \frac{1}{2}$$

$$x_1 + (0)x_3 + (0)x_4 \leq 2 + \frac{1}{2}$$

$$x_1 + (0)x_3 + (0)x_4 \leq 2$$

$$x_1 + (0)x_3 + (0)x_4 + s = 2$$

$$\begin{array}{l} + \\ - \end{array} \quad \frac{1}{4}x_3 + \frac{1}{4}x_4 - s = \frac{1}{2}$$

$$\frac{1}{4}x_3 + \frac{1}{4}x_4 \geq \frac{1}{2}$$

La solution de base ne vérifie pas cette inégalité
Elle sera exclue par cette inégalité

Autre coupe

base	x_1	x_2	x_3	x_4	
x_1	1		$\frac{1}{4}$	$\frac{1}{4}$	$\frac{5}{2}$
x_2		1	$\frac{3}{4}$	$-\frac{1}{4}$	$\frac{3}{2}$
			$-\frac{5}{4}$	$-\frac{1}{4}$	$-\frac{13}{2} + z$

fractionnaire

$$x_2 + \frac{3}{4}x_3 - \frac{1}{4}x_4 = \frac{3}{2}$$

$$x_2 + (0 + \frac{3}{4})x_3 + (-1 + \frac{3}{4})x_4 = 1 + \frac{1}{2}$$

$$x_2 + (0)x_3 + (-1)x_4 \leq 1 + \frac{1}{2}$$

$$x_2 + (0)x_3 + (-1)x_4 \leq 1$$

$$x_2 + (0)x_3 + (-1)x_4 + s = 1$$

+

-

$$\frac{3}{4}x_3 + \frac{3}{4}x_4 - s = \frac{1}{2}$$

$$\frac{3}{4}x_3 + \frac{3}{4}x_4 \geq \frac{1}{2}$$

Formule générale de la coupe:

$$\sum_{j \text{ t.q. } x_j \text{ hors-base}} f_{ij} x_j \geq f_i$$

Algorithme des coupes de Gomory

base	x_1	x_2	x_3	x_4	
x_1	1		$\frac{1}{4}$	$\frac{1}{4}$	$\frac{5}{2}$
x_2		1	$\frac{3}{4}$	$-\frac{1}{4}$	$\frac{3}{2}$
			$-\frac{5}{4}$	$-\frac{1}{4}$	$-\frac{13}{2} + z$

Coupe à ajouter

$$\frac{1}{4}x_3 + \frac{1}{4}x_4 \geq \frac{1}{2} \rightarrow \frac{1}{4}x_3 + \frac{1}{4}x_4 - s = \frac{1}{2} \rightarrow -\frac{1}{4}x_3 - \frac{1}{4}x_4 + s = -\frac{1}{2}$$

base	x_1	x_2	x_3	x_4	s	
x_1	1		$\frac{1}{4}$	$\frac{1}{4}$		$\frac{5}{2}$
x_2		1	$\frac{3}{4}$	$-\frac{1}{4}$		$\frac{3}{2}$
s			$-\frac{1}{4}$	$-\frac{1}{4}$	1	$-\frac{1}{2}$
			$-\frac{5}{4}$	$-\frac{1}{4}$		$-\frac{13}{2} + z$



Solution de base non réalisable car $s < 0$

Algorithme dual du simplexe

base	x_1	x_2	x_3	x_4	s	
x_1	1		$\frac{1}{4}$	$\frac{1}{4}$		$\frac{5}{2}$
x_2		1	$\frac{3}{4}$	$-\frac{1}{4}$		$\frac{3}{2}$
s			$-\frac{1}{4}$	$-\frac{1}{4}$	1	$-\frac{1}{2}$
			$-\frac{5}{4}$	$-\frac{1}{4}$		$-\frac{13}{2} + z$

→ s sort

Qui rentre?

$$(-\frac{5}{4}x_3 - \frac{1}{4}x_4 = -\frac{13}{2} + z) + \rho \times (-\frac{1}{4}x_3 - \frac{1}{4}x_4 + s = -\frac{1}{2})$$

$$(-\frac{5}{4} + \rho \times (-\frac{1}{4}))x_3 + (-\frac{1}{4} + \rho \times (-\frac{1}{4}))x_4 + \rho \times s = -\frac{13}{2} + \rho \times (-\frac{1}{2}) + z$$

les coûts réduits sont ≤ 0 :

$$-\frac{5}{4} + \rho \times (-\frac{1}{4}) \leq 0, -\frac{1}{4} + \rho \times (-\frac{1}{4}) \leq 0, \rho \leq 0,$$

$$\text{ce qui donne } (-\frac{5}{4}) / (-\frac{1}{4}) \leq \rho, (-\frac{1}{4}) / (-\frac{1}{4}) \leq \rho, \rho \leq 0.$$

Pour qu'un coût réduit s'annule il faut prendre :

$$\rho = \max\{(-\frac{5}{4}) / (-\frac{1}{4}), (-\frac{1}{4}) / (-\frac{1}{4})\} = (-\frac{1}{4}) / (-\frac{1}{4})$$

ce qui correspond à la variable x_4 . La variable x_4 rentre en base.

Algorithme dual du simplexe (suite)

base	x_1	x_2	x_3	x_4	s	
x_1	1		$\frac{1}{4}$	$\frac{1}{4}$		$\frac{5}{2}$
x_2		1	$\frac{3}{4}$	$-\frac{1}{4}$		$\frac{3}{2}$
s			$-\frac{1}{4}$	$-\frac{1}{4}$	1	$-\frac{1}{2}$
			$-\frac{5}{4}$	$-\frac{1}{4}$		$-\frac{13}{2} + z$

s sort $\longrightarrow$

x_4 rentre en base $\longrightarrow$

base	x_1	x_2	x_3	x_4	s	
x_1	1		0	0	1	2
x_2		1	1	0	-1	2
x_4			1	1	-4	2
			-1	0	-1	$-6 + z$

Solution ≥ 0 + Coûts réduits ≤ 0 et on maximise $\Rightarrow$ STOP

Programmation linéaire en nombres entiers: modélisation

Un exemple:

n clients C_i $i=1, \dots, n$ munis de poids $w_i > 0$. C_i de coordonnées c_{1i} et c_{2i}

p magasins M_j $j=1, \dots, p$. M_j de coordonnées m_{1j} et m_{2j}

Un nouveau magasin M concurrent veut s'implanter et maximiser le poids total des clients qu'il aura capturés

Un client i est capturé quand

$$d(C_i, M) \leq R_i = \min\{d(C_i, M_j) : j=1, \dots, p\}$$

La distance du client i au nouveau magasin $\leq$ à la distance du client i au magasin déjà existant le plus proche

Modèle

Variables $\left\{ \begin{array}{l} y_i=1 \text{ si client } C_i \text{ est capturé , 0 sinon} \\ x_1, x_2 \text{ coordonnées du nouveau magasin} \\ d_i \text{ distance du client } C_i \text{ au nouveau magasin mesurée par la norme 1} \end{array} \right.$

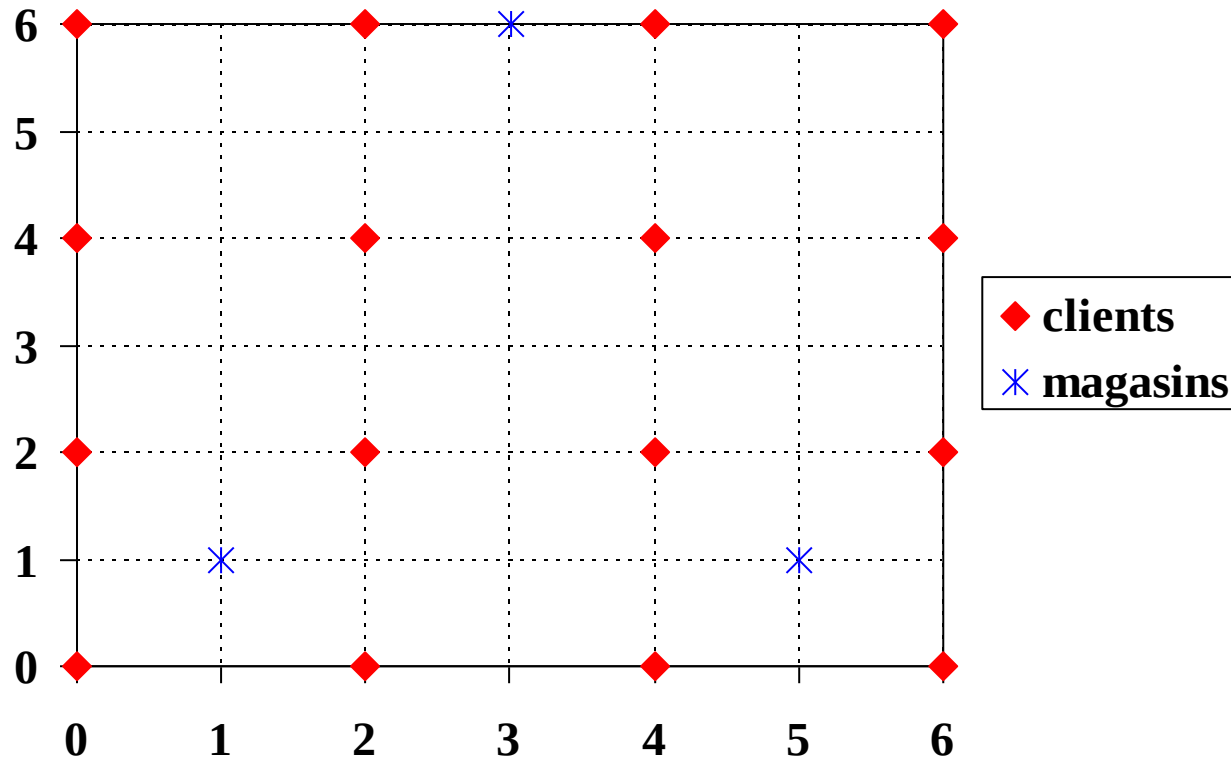
$$\begin{array}{ll} \max & \sum_{i=1,n} w_i y_i \\ \text{s.c.} & \left[\begin{array}{l} d_i - h \times (1 - y_i) \leq R_i \quad (1) \\ d_i \geq x_1 - c_{1i} + x_2 - c_{2i} \quad (2.1) \\ d_i \geq x_1 - c_{1i} - x_2 + c_{2i} \quad (2.2) \\ d_i \geq -x_1 + c_{1i} + x_2 - c_{2i} \quad (2.3) \\ d_i \geq -x_1 + c_{1i} - x_2 + c_{2i} \quad (2.4) \\ y_i \in \{0,1\} \end{array} \right] \quad i = 1, \dots, n \end{array}$$

(1) $y_i=0 \Rightarrow d_i - h \leq R_i$ h constante suffisamment grande $\Rightarrow$ contrainte inopérante

(1) $y_i=1 \Rightarrow d_i \leq R_i$ distance du client C_i au magasin $\leq$ min des distances
aux autres magasins

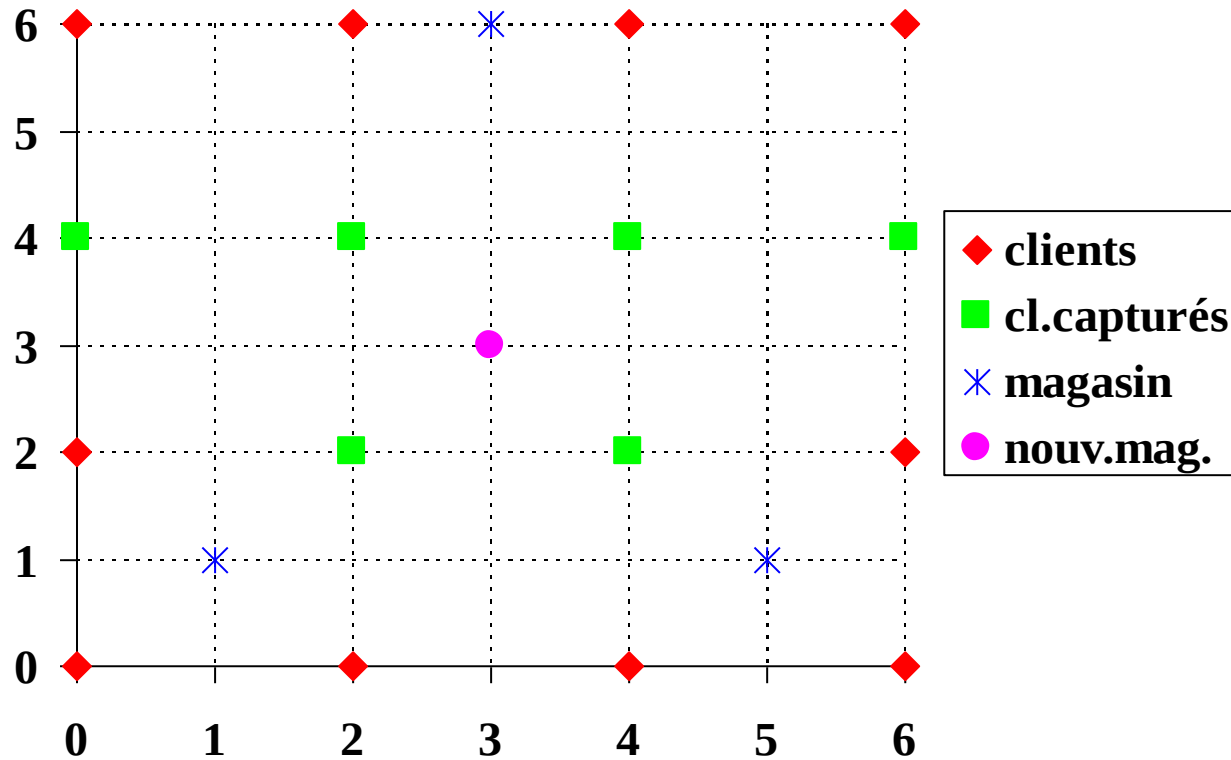
(2.1) à (2.4) $\Leftrightarrow d_i \geq |x_1 - c_{1i}| + |x_2 - c_{2i}|$

Exemple 1: 16 clients de poids 1 chacun
3 magasins déjà implantés

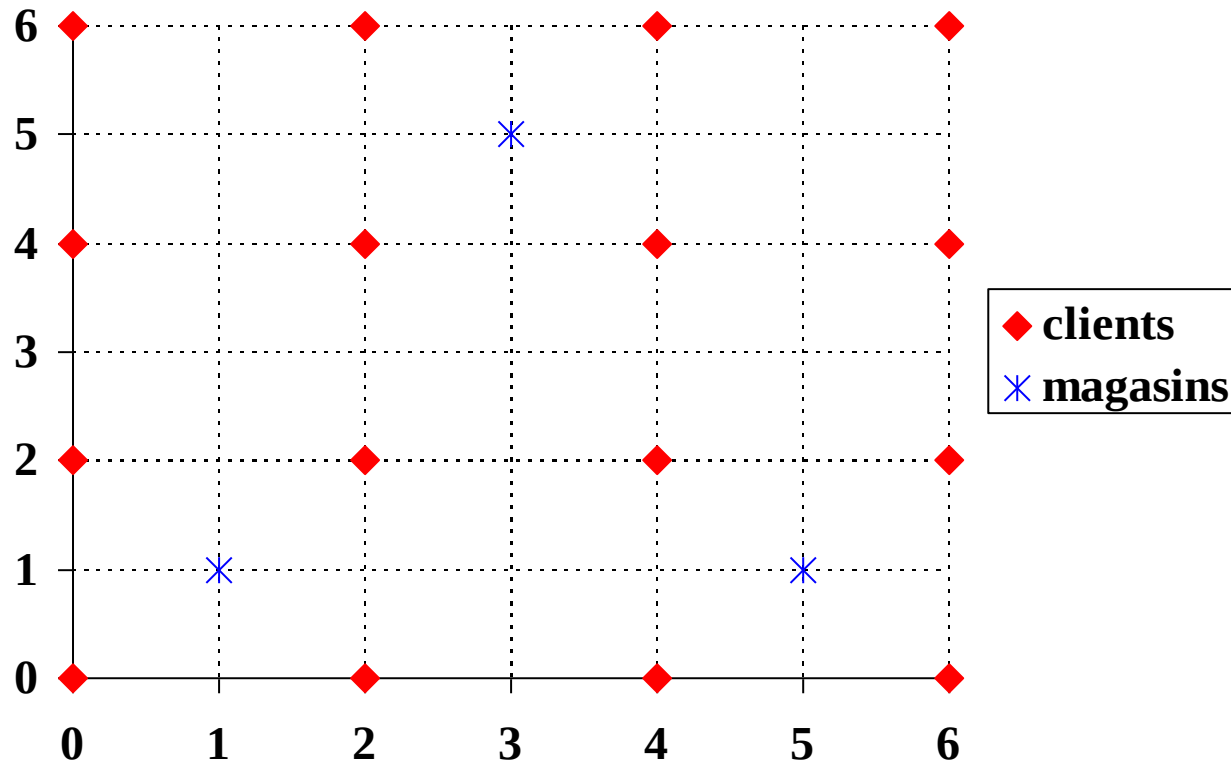


La distance max. entre deux points est $h=12$

Résultat: 6 clients capturés



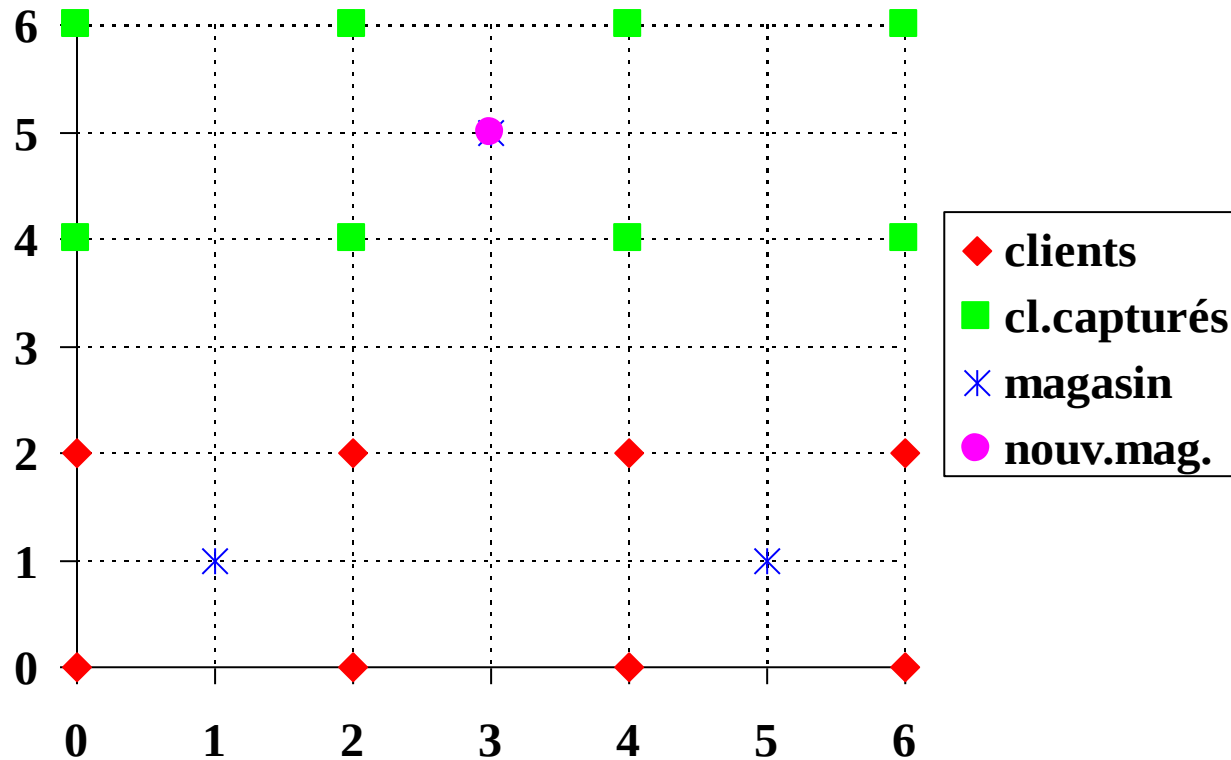
Exemple 2: 16 clients de poids 1 chacun
3 magasins déjà implantés



La distance max. entre deux points est $h=12$

Résultat: 8 clients capturés

Superposition du nouveau magasin sur un ancien. Pas très réaliste



Modèle avec une distance min. S entre le nouveau magasin et les anciens

$$\begin{aligned}
 & \max \sum_{i=1,n} w_i y_i \\
 & \text{s.c.} \left\{ \begin{array}{l} \left[\begin{array}{l} d_i - h \times (1 - y_i) \leq R_i \quad (1) \\ d_i \geq x_1 - c_{1i} + x_2 - c_{2i} \quad (2.1) \\ d_i \geq x_1 - c_{1i} - x_2 + c_{2i} \quad (2.2) \\ d_i \geq -x_1 + c_{1i} + x_2 - c_{2i} \quad (2.3) \\ d_i \geq -x_1 + c_{1i} - x_2 + c_{2i} \quad (2.4) \\ y_i \in \{0,1\} \end{array} \right] i = 1, \dots, n \\ \left[\begin{array}{l} x_1 - m_{1j} + x_2 - m_{2j} \geq -h + (h + S)z_{1j} \quad (3.1) \\ x_1 - m_{1j} - x_2 + m_{2j} \geq -h + (h + S)z_{2j} \quad (3.2) \\ -x_1 + m_{1j} + x_2 - m_{2j} \geq -h + (h + S)z_{3j} \quad (3.3) \\ -x_1 + m_{1j} - x_2 + m_{2j} \geq -h + (h + S)z_{4j} \quad (3.4) \\ z_{1j} + z_{2j} + z_{3j} + z_{4j} = 1 \quad (3.5) \\ z_{1j}, z_{2j}, z_{3j}, z_{4j} \in \{0,1\} \end{array} \right] j = 1, \dots, p \end{array} \right.
 \end{aligned}$$

$$(3.1) \text{ à } (3.5) \Leftrightarrow |x_1 - m_{1j}| + |x_2 - m_{2j}| \geq S$$

distance min. $S=1$ entre le nouveau magasin et les anciens
Résultat: 7 clients capturés

