Introduction à la Programmation Linéaire en Nombres Entiers

Leo Liberti, Ruslan Sadykov

LIX, École Polytechnique

liberti@lix.polytechnique.fr

sadykov@lix.polytechnique.fr

Qu'est-ce qu'on va faire et pourquoi ?

Les avantages de la Programmation Linéaire en Nombres Entiers (PLNE)

- On peux modéliser plus de problèmes comme PLNE que comme Programmes Linéaires
- Il y a des méthodes pour résoudre les PLNE, qui sont souvent efficaces en pratique
- Il y a des solveurs qu'on peut utiliser pour appliquer ces méthodes rapidement

Les buts principaux du cours

- Savoir modéliser des problèmes
- Savoir utiliser des solveurs
- Savoir comment les solveurs marchent à l'intérieur

Contenu

- Exemples des Programmes Linéaires en Nombres Entiers
- La méthode de coupes (*Cutting planes method*)
- La méthode de recherche arborescente par séparation et évaluation (*Branch-and-Bound*)
- La méthode que combine les deux dernières (*Branch-and-Cut*)

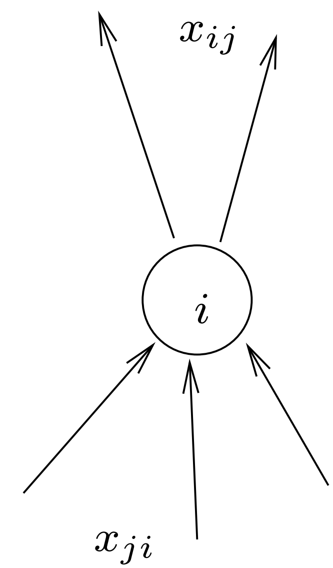
Problèmes formulables en PLNE

- Problèmes avec entrées/sorties discrètes : production d'objets, etc.
- Problèmes avec conditions logiques : ajout de variables entières avec des constraintes supplémentaires.
- Problèmes combinatoires : séquençage, allocation de ressources, emplois du temps, etc.
- Une partie des problèmes non-linéaires.
- Problèmes de réseaux, problèmes de graphes.

Le problème de flot maximum

On a un graphe dirigé $G = (V, A)$ (un réseau) avec une source s, une destination t, et des capacités entières u_{ij} sur chaque arrête (i, j). On doit déterminer la quantité maximum de flot entier de la matière qui peut circuler sur le réseau de s à t. Les variables sont $x_{ij} \in \mathbb{Z}_+$, définées pour chaque arrête (i, j) du graphe, représentent les nombres des unités du flot passées par (i, j).

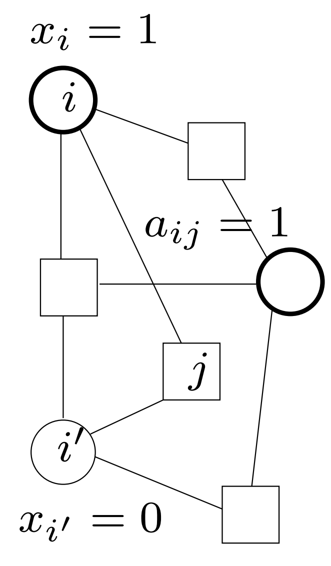
$$\left.\begin{array}{lcc}
\max_x & & \displaystyle\sum_{(s,i)\in A} x_{si} \\
\forall\, i \le V, \begin{array}{c} i \neq s \\ i \neq t \end{array} & & \displaystyle\sum_{(i,j)\in A} x_{ij} = \sum_{(j,i)\in A} x_{ji} \\
\forall (i,j) \in A & & 0 \le x_{ij} \le u_{ij} \\
\forall (i,j) \in A & & x_{ij} \in \mathbb{Z}_+
\end{array}\right\}$$



Le problème de recouvrement

On a n villes et m régions où on peut construire une installation. Le coût de construction d'une installation dans la région $i \leq m$ est f_i. Soit $a_{ij} = 1$ si est seulement si l'installation dans la région i peut servir la ville j. On doit minimiser le coût de construction des installations sous une contrainte : chaque ville doit être sérvit par au moins une installation construite. Les variables sont $x_i \in \{0, 1\}$, définées pour chaque ville i, indiquent si l'installation i doit être construite ou pas.

$$\left.\begin{array}{rrcl} \min_x & \sum_{i=1}^{m} f_i x_i & & \\ \forall\, j \leq n & \sum_{i=1}^{m} a_{ij} x_i & \geq & 1 \\ \forall\, i \leq m & x_i & \in & \{0,1\} \end{array}\right\}$$



Définitions

- Programme Linéare en Nombres Entiers :

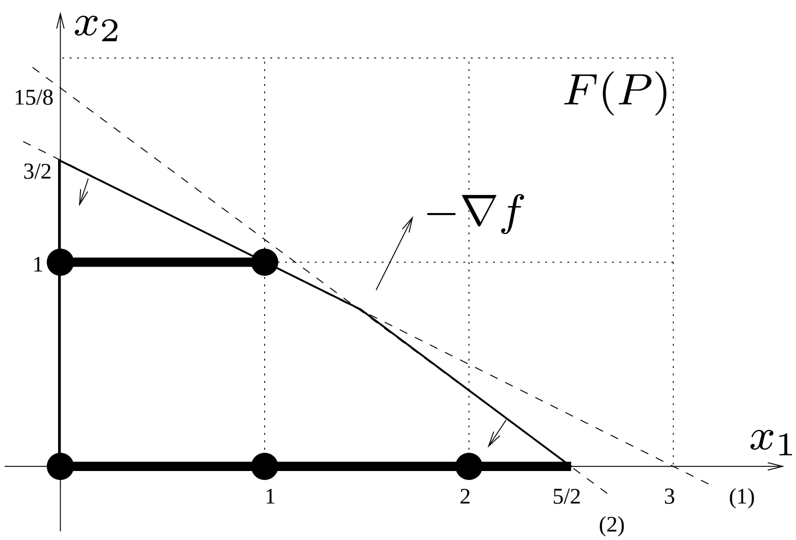
$$\left.\begin{array}{rl} \min_x & c^\mathsf{T} x + d^\mathsf{T} y \\ \text{s.t.} & Ax + By \leq b \\ & x \geq 0, y \geq 0, \\ & x \in \mathbb{Z}^n_+ \end{array}\right\} [P] \qquad (1)$$

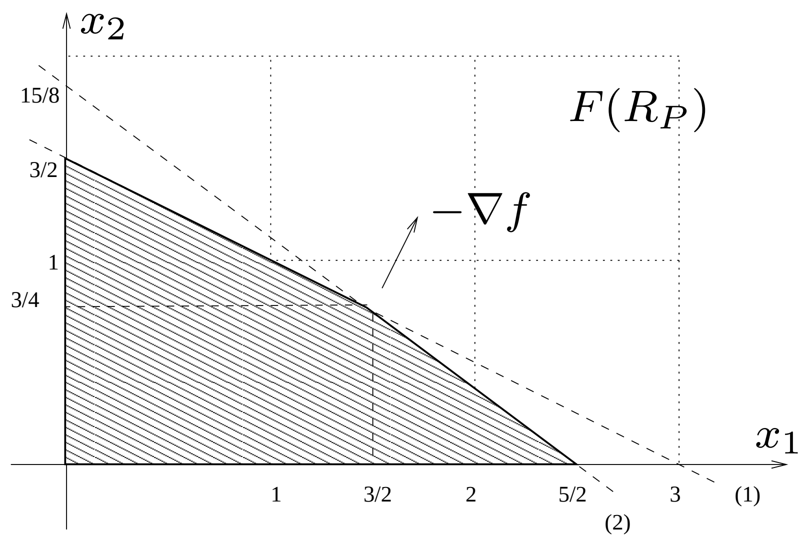
- La *relaxation linéaire* (ou *continue*) R_P de P est obtenue par relâchement (enlèvement) des contraintes d'intégralité ($x \in \mathbb{Z}^n_+ \;\rightarrow\; x \geq 0$).
- Soit $F(P)$ la région des solutions possibles de P : on a $F(P) \subseteq F(R_P)$.
- Soit (x^*, y^*) une solutoin optimale de P et soit $(\bar{x}, \bar{y})$ une solution optimale de R_P ; alors $c^\mathsf{T}\bar{x} + d^\mathsf{T}\bar{y} \leq c^\mathsf{T}x^* + d^\mathsf{T}y^*$: la valeur optimale de R_P est une *borne inférieur* pour P.

Un exemple simple

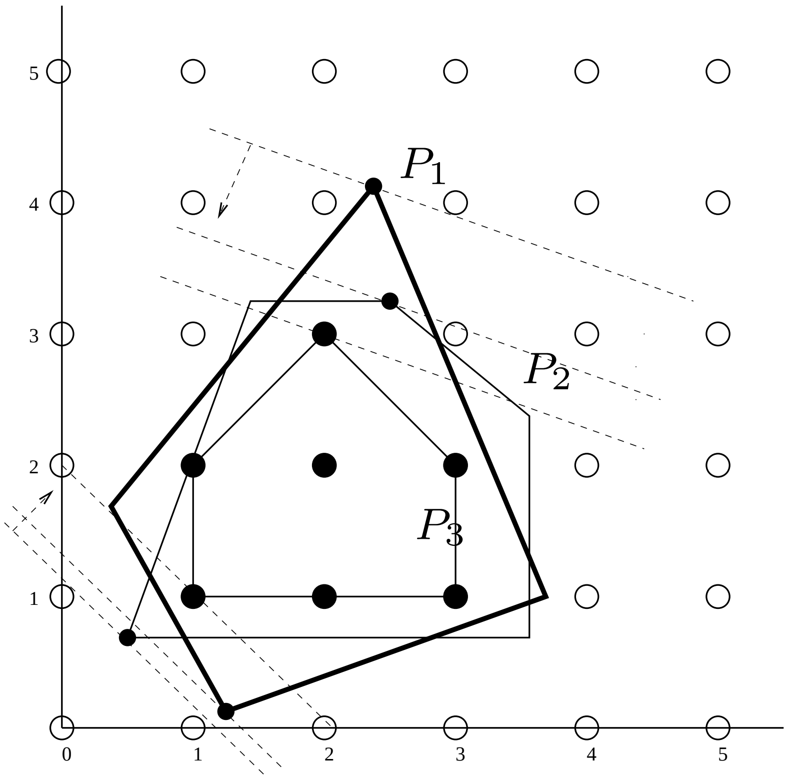
Consider example :

$$\left.\begin{array}{rl} \min f = & -3x_1 - 5x_2 \\ & x_1 + 2x_2 \leq 3 \\ & 6x_1 + 8x_2 \leq 15 \\ & x_1 \in \mathbb{R}_+,\ x_2 \in \mathbb{Z}_+ \end{array}\right\}$$





Bonnes formulations



Plus $F(R_P)$ est petit, plus la borne inférieur produite par R_P est proche de la valeur optimale de P. Comme $F(R_{P_3}) \subset F(R_{P_2})$ et $F(R_{P_3}) \subset F(R_{P_2})$, la formulation P_3 est meilleur que P_1 et P_2.

Ici P_3 est la meilleur formulation (formulation idéale). On dit que R_{P_3} est *l'enveloppe convexe* de P_1 et P_2

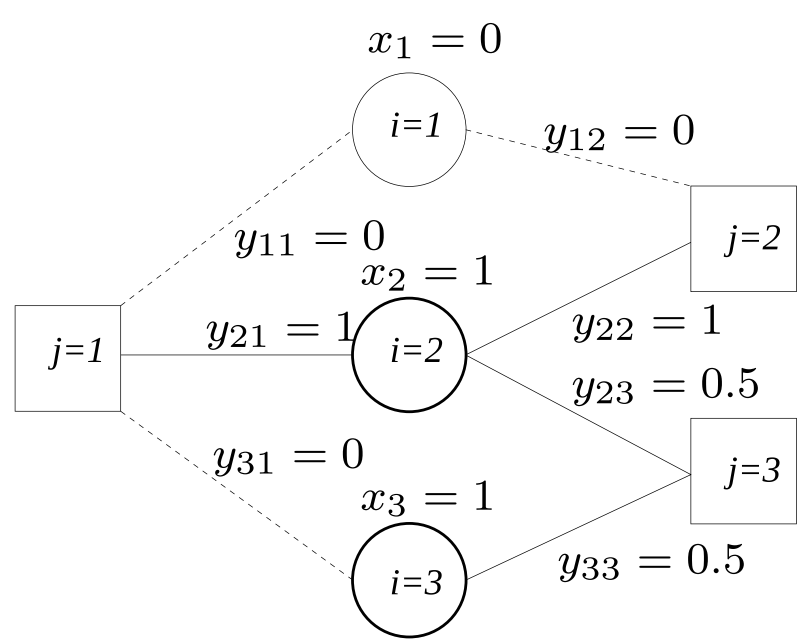
Formellement, si $P = \{x^1, \ldots, x^t\}$, alors

$$conv(P) = \{x : x = \sum_{i=1}^{t} \lambda_i x^i,\ \sum_{i=1}^{t} \lambda_i = 1,\ \lambda_i \geq 0, \forall i = 1, \ldots, t\}.$$

Problème de localisation des installations

Similaire au problème de recouvrement. Ici on ajoute les coûts de transportation c_{ij} de la demande de la ville j à partir de l'installation i. Les variables additionelles $y_{ij} \in \mathbb{R}_+$ représentent la fraction de la demande de la ville j servie par l'installation i.

$$\left.\begin{array}{c} \min\limits_{x,y} \sum\limits_{i=1}^{m} f_i x_i + \sum\limits_{i=1}^{m}\sum\limits_{j=1}^{n} c_{ij} y_{ij} \\ \forall j \le n, \quad \sum\limits_{i=1}^{m} y_{ij} = 1 \\ \forall i \le m, \quad \sum\limits_{j=1}^{m} y_{ij} \le n x_i \\ \forall i \le m, \forall j \le n, \quad y_{ij} \ge 0, \\ \forall i \le m, \quad x_i \in \{0,1\}. \end{array}\right\}$$



Problème LI (II)

On peut changer les contraintes

$$\forall i \leq m, \quad \sum_{j=1}^{n} y_{ij} \leq n x_i \quad [P_1]$$

par

$$\forall i \leq m, \forall j \leq n, \quad y_{ij} \leq x_i. \quad [P_2]$$

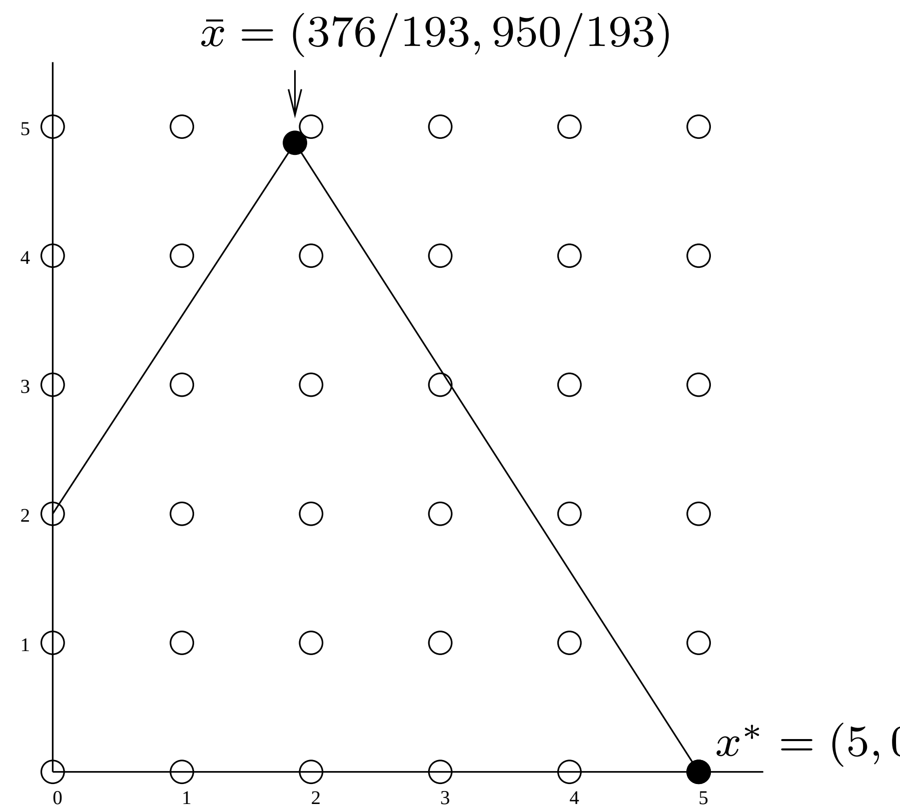
Formulation P_2 is meilleur que P_1 as $F(R_{P_2}) \subset F(R_{P_1})$.
Preuve. On a $(x,y) \in F(R_{P_2}) \Rightarrow (x,y) \in F(R_{P_2})$. Puis soit $m = 1$, $n = 2$, $x'_1 = 0.5$, $y'_1 = (1,0)$. Alors $(x', y') \in F(R_{P_1})$ et $(x', y') \notin F(R_{P_2})$.

Heuristique d'arrondissement

Il y a une forte rélation entre une PLNE et sa relaxation continue.

Est-ce que c'est bon juste arrondir la solution optimale de la relaxation continue ? Pas toujours. On envisage le PLNE suivant :

$$\max 1.00x_1 + 0.64x_2$$
$$50x_1 + 31x_2 \leq 250$$
$$3x_1 - 2x_2 \geq -4$$
$$x_1, x_2 \in \mathbb{Z}_+$$

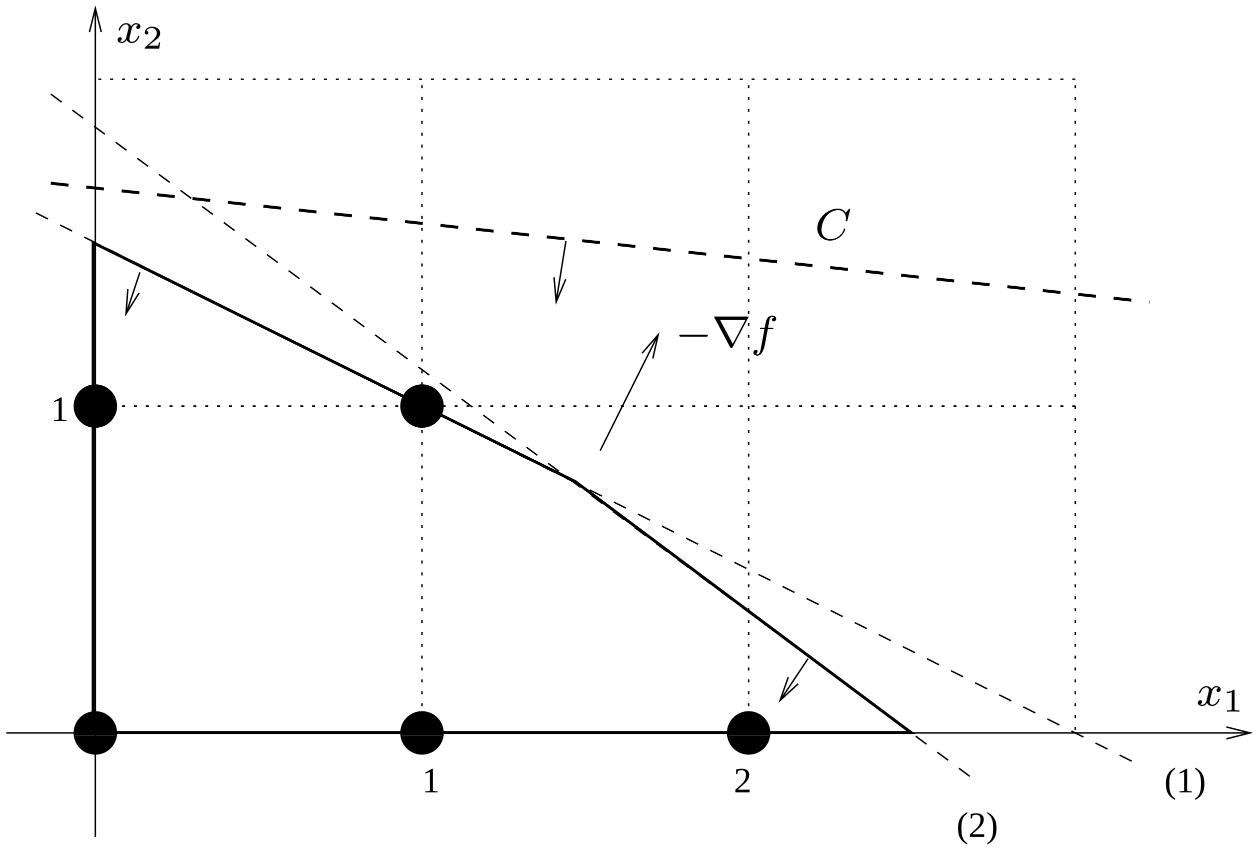


Les idées algorithmiques principales

- Si on peut dire à priori que $\bar{x} \in \mathbb{Z}^n$, alors on peut juste résoudre R_P au lieu de P (*la propriété d'unimodularité totale*).
- On peut ajouter des contraintes à P pour obtenir P' tel que $\bar{x}' \in \mathbb{Z}^n$ (*L'algorithme de coupes*).
- On peut énumérer les solutions de façon intelligente (*Branch-and-Bound*).
- On peut combiner l'ajout des contraintes et l'énumération (*Branch-and-Cut*).
- Les solveurs modernes (comme *Cplex*) utilisent *Branch-and-Cut* à l'intérieur.

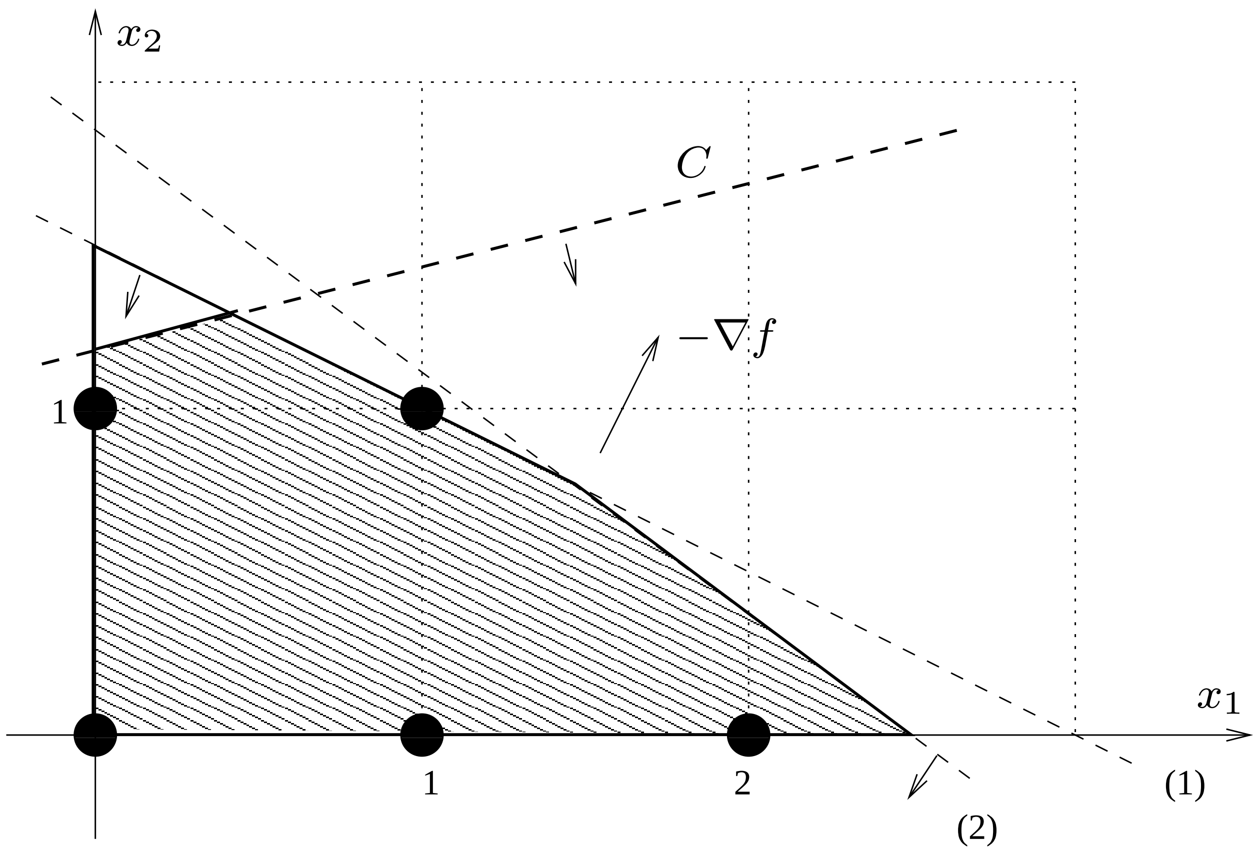
Les plans coupants : définitions I

Une contrainte $C \equiv \pi^\mathsf{T} x \leq \pi_0$ est *valide* pour P si $\forall\, x' \in F(P)\ (\pi^\mathsf{T} x' \leq \pi_0)$



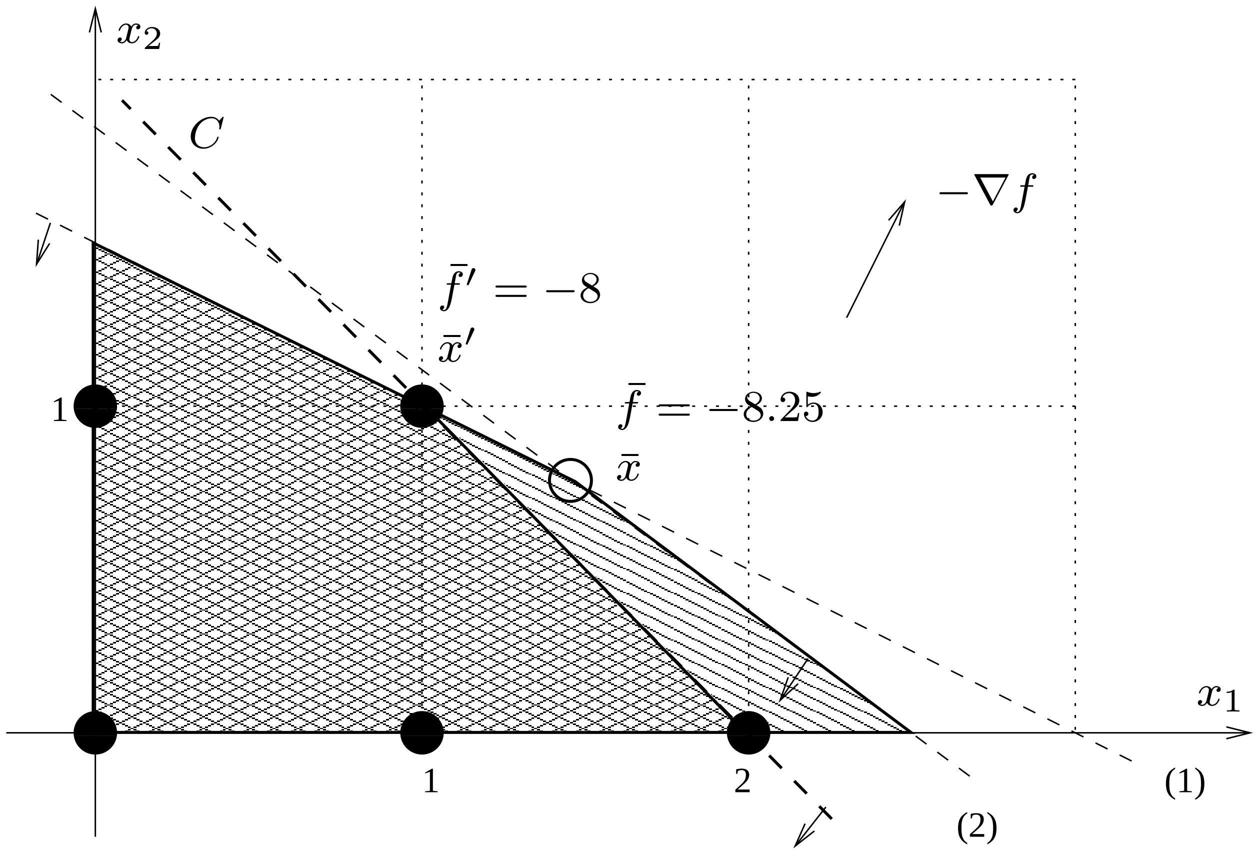
Les plans coupants : définitions II

Soit P' le problème P avec une contrainte valide C ajoutée.
C est le *plan coupant* pour P si $F(R_{P'}) \subset F(R_P)$



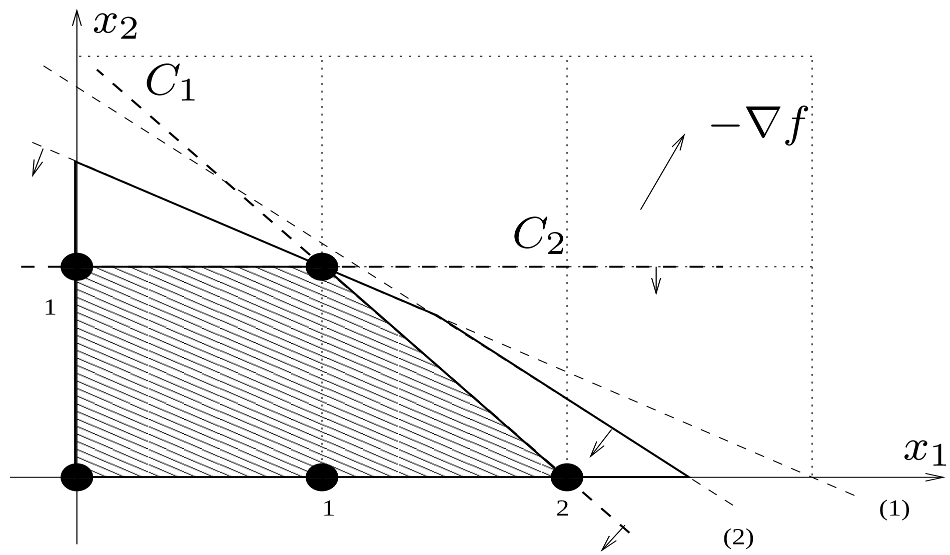
Les plans coupants : définitions III

Un plan coupant $C: \ \pi^{\mathsf{T}} x \leq \pi_0$ est une *coupe valide* pour $\bar{x} \in R_P$ si $\pi^{\mathsf{T}} \bar{x} > \pi_0$.



L'enveloppe convexe

- Pour obtenir la description de l'enveloppe convexe de $F(P)$, on a besoin un nombre finit de contraintes valides pour P.



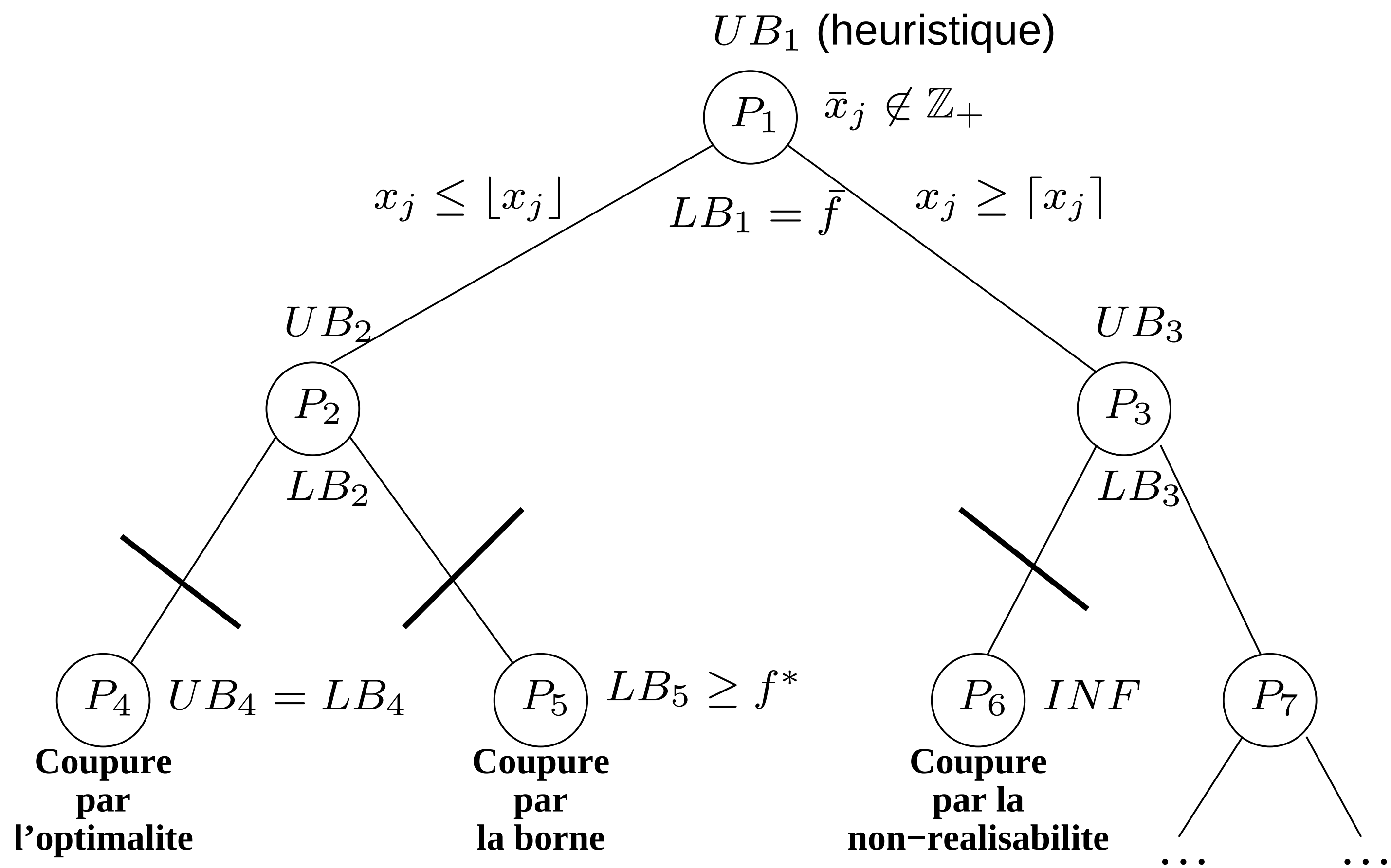
- Le problème de calculation de l'enveloppe convexe pour $F(P)$ est un problème plus difficile en général que P.

L'algorithme de coupes

- Stratégie :
  1. On résout R_P et on obtient la solution $\bar{x}$
  2. Si $\bar{x} \in \mathbb{Z}^n$ le problème est résolu, sortie.
  3. On construit une coupe valide C pour $\bar{x}$ et P
  4. On ajoute la contrainte C à la formulation P
  5. Revenir à l'étape 1
- L'étape la plus importante est 3 (*le problème de séparation*).
- Les algorithmes de coupes peuvent dépendre de la structure du problème ou peuvent être généraux (*l'algorithme de coupe de Gomory*).

Branch-and-Bound I

On utilise l'approche "diviser-pour-reigner". Si on peut pas résoudre un problème, on le divise en sous-problèmes plus simples.



Branch-and-Bound II

1. On initialise la liste des problèmes $L = \{P\}$, la meilleur valeur trouvée de la fonction objective $f^* = \infty$, $x^* =$ “infeasible”
2. Si $L = \emptyset$, on termine, x^* est une solution optimale
3. On choisit un sous-problème Q de L, $L = L \setminus \{Q\}$
4. On résout R_Q, $\bar{x}$ est la solution avec la valeur $\bar{f}$
5. Si $F(R_Q) = \emptyset$, retour à 2 (coupure par la non-réalisabilité)
6. Si $\bar{f} \geq f^*$, Q ne contient pas une solution meilleur que f^*, retour à 2 (coupure par la borne)
7. Si $\bar{x} \in \mathbb{Z}_+$ et $\bar{f} < f^*$: on met à jour $x^* = \bar{x}$, $f^* = \bar{f}$, retour à 2 (coupure par l’optimalité)
8. On choisi une variable $\bar{x}_j$, $\bar{x}_j \notin \mathbb{Z}$, et génère deux sous-problèmes de Q en ajoutant les constraintes $x_j \leq \lfloor \bar{x}_j \rfloor$ et $x_j \geq \lceil \bar{x}_j \rceil$ respectivement, on les ajoute à L, retour à 2.

Branch-and-Bound III

L'algorithme admet plusieurs variations.

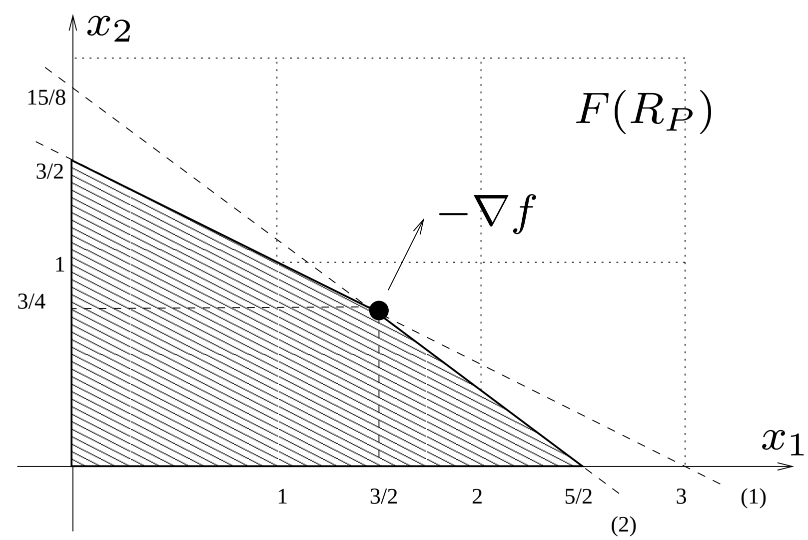
- Comment choisit-on un sous-problème Q dans la liste L (l'étape 3) ?
- Comment choisit-on une variable fractionnaire x_j (l'étape 8) ?
- Il n'y a pas de “meilleur réponse”, ça dépend de la structure du problème.
- Les solveurs permettent à l'utilisateur de préciser quelle variation de l'algorithme doit être utilisée. Il y a toujours la variation par défaut qui convient dans la majorité des cas.

BB exemple I

On envisage un exemple simple :

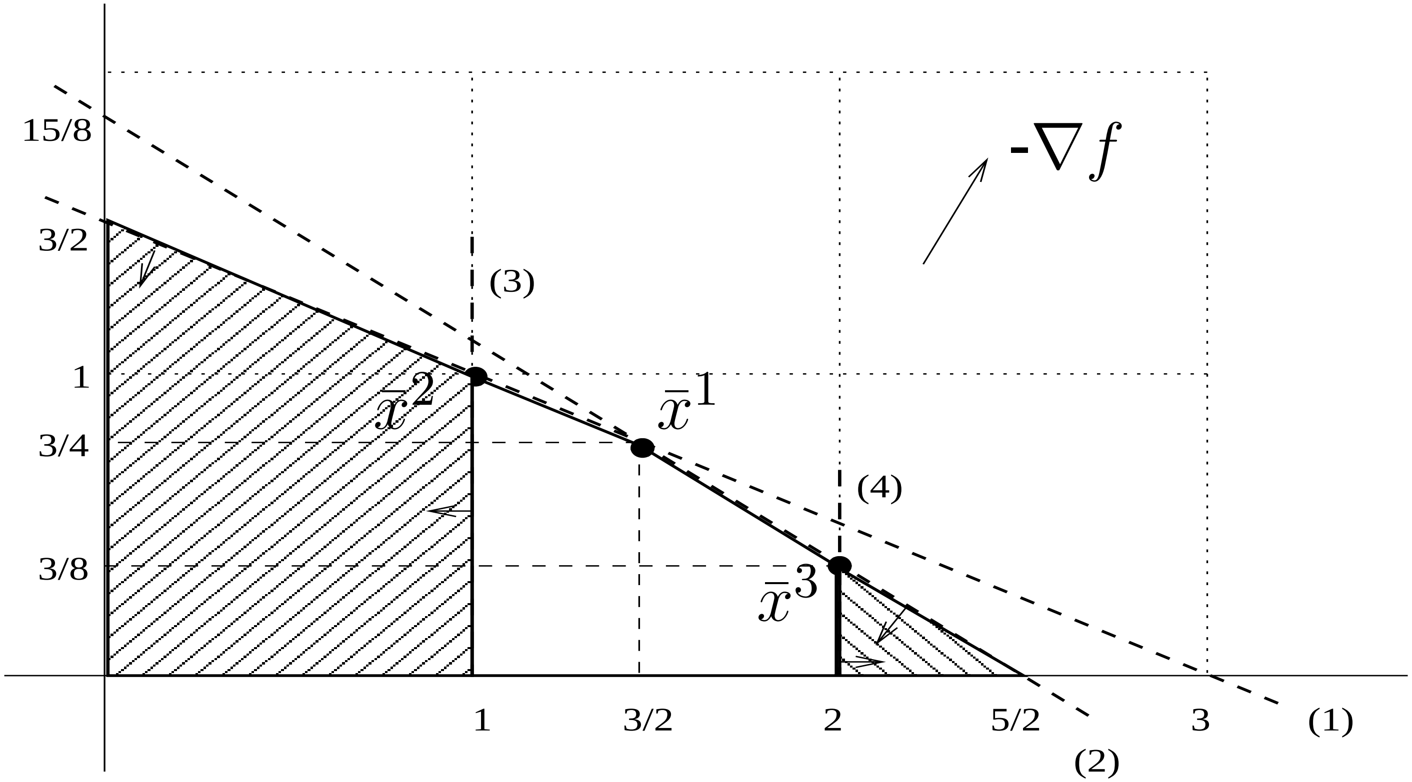
$$\left.\begin{array}{rcl} \min \quad -3x_1 - 5x_2 & & \\ x_1 + 2x_2 & \leq & 3 \\ 6x_1 + 8x_2 & \leq & 15 \\ x_1, x_2 & \in & \mathbb{Z}_+ \end{array}\right\}$$

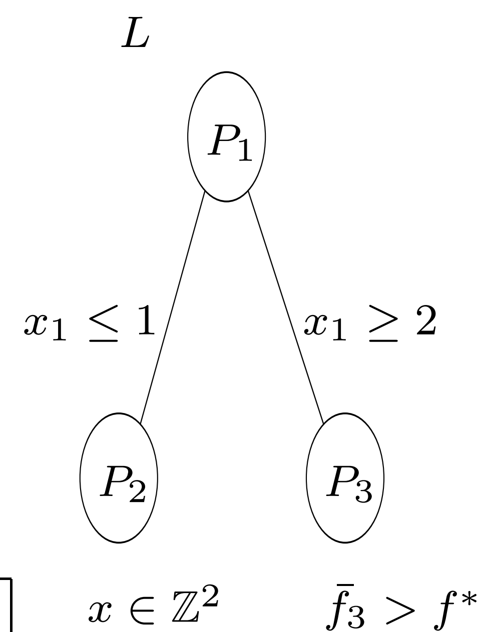
La solution de R_P est $\bar{x} = (1.5, 0.75)$ avec $\bar{f} = -8.25$



BB exemple II

$\bar{x}^i =$ la solution de R_{P_i}, $\bar{f}_i =$ la valeur optimale de R_{P_i}, $\forall i$





P_2	P_1	P_3
$\bar{x}^2 = (1,1)$ $\bar{f}_2 = -8$	$\bar{x}^1 = (1.5, 0.75)$ $\bar{f}_1 = -8.25$	$\bar{x}^3 = (2, 0.375)$ $\bar{f}_3 = -7.875$

Branch-and-Cut

- Dans l'algorithme Branch-and-Bound, avant la division en sous-problèmes, on génère des coupes valides pour la solution fractionnaire courante $\bar{x}$.
- Généralement, les coupes sont générées jusqu'à ce que la valeur $\bar{f}$ de la fonction objective n'augmente pas beaucoup.
- Par défaut, les solveurs utilisent les coupes générales.
- Les classes des coupes générales les plus utilisées : *Gomory cuts*, *Mixed-Integer Rounding (MIR) cuts*, *[flow] cover cuts*.

Les inégalités de Gomory

- Soit $P = \{x \in \mathbb{Z}_+^n : \; Ax \leq b\}$, A est une matrice $m \times n$ avec les colonnes $(a_1, \ldots, a_n)$, et $u \in \mathbb{R}_+^m$.
- $\displaystyle\sum_{j=1}^{n} ua_j x_j \leq ub$ est valide pour P, comme $u \geq 0$;
- $\displaystyle\sum_{j=1}^{n} \lfloor ua_j \rfloor x_j \leq ub$ est valide pour P, comme $x \geq 0$;
- $\displaystyle\sum_{j=1}^{n} \lfloor ua_j \rfloor x_j \leq \lfloor ub \rfloor$ est valide pour P, comme x est entier.
- En utilisant cette procédure, on peut générer **toutes** les inégalités (contraintes) valides pour un PLNE.

Les inégalités de couverture

- Soit $P = \left\{x \in \{0,1\}^n : \sum_{j=1}^n a_j x_j \leq b\right\}$, $a_j \geq 0$, $\forall j \leq n$, $b \geq 0$, $N = \{1, 2, \ldots, n\}$.
- Un ensemble $C \subseteq N$ est une *couverture* si $\sum_{j \in C} a_j > b$.
- Si $C \subseteq N$ est une couverture, alors l'inégalité de couverture

$$\sum_{j \in C} x_j \leq \mid C \mid -1$$

est valide pour P.

Quelques solveurs PLNE existants

Payants

- ILOG Cplex (`www.ilog.com`)
- Xpress-Optimizer (`www.dashoptimization.com`)
- LINDO (`www.lindo.com`)

Gratuits

- COIN-OR (`www.coin-or.org`)
- GLPK (`www.gnu.org/software/glpk`)

Les ouvrages pour le cours

- C. Papadimitriou, K. Steiglitz, *Combinatorial Optimization : Algorithms and Complexity*, Dover, New York, 1998
- L. Wolsey, *Integer Programming*, John Wiley & Sons, Inc, New York, 1998.