

Programmation Linéaire en Nombres Entiers

Cours d'Optimisation Combinatoire

Domaine Conception et Opération des Systèmes Spatiaux

ISAE-SUPAERO

Christian Artigues

LAAS-CNRS

2 septembre 2015

- 1 Programmation linéaire en nombres entiers (PLNE) : introduction et définition
- 2 Modélisation : formulation en PL(NE) d'un problème d'optimisation
 - Définition d'une formulation correcte
 - Qualité et comparaison des formulations
 - Unimodularité
 - Modélisation de fonctions élémentaires
 - Modélisation de problèmes typiques d'optimisation combinatoire
- 3 Méthodes de résolution
 - Méthodes de coupes
 - Méthode de séparation et évaluation
- 4 Sources et compléments
- 5 Références des applications spatiales

- La programmation linéaire désigne un paradigme de résolution de problèmes d'optimisation où toutes les contraintes sont linéaires.
- En programmation linéaire (PL), toutes les variables de décision sont continues. Le problème d'optimisation correspondant est polynomial.
- En programmation linéaire en nombres entiers (PLNE), il existe des variables de décision entières. Le problème d'optimisation correspondant est NP-difficile.
- De nombreux solveurs sur étagère permettent de résoudre des programmes linéaires (en nombres entiers) de manière efficace.
- Il est donc intéressant en pratique de modéliser un problème réel (même s'il n'apparaît pas comme linéaire à première vue) comme un PL ou un PLNE puis d'utiliser un de ces solveurs pour le résoudre.

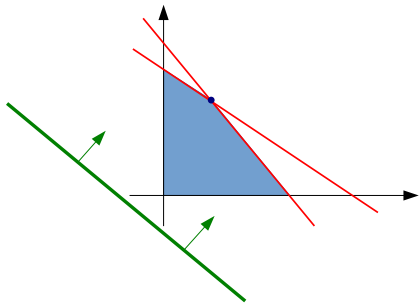
Objectifs de cette partie du cours

- 1 Apprendre à modéliser un problème d'optimisation combinatoire comme un PL(NE).
- 2 Connaître les méthodes de résolution de base pour résoudre un PLNE.

$$(PL) \left\{ \begin{array}{l} \text{Maximiser} \quad \sum_{i=1}^n c_i x_i \\ \text{sous les contraintes} \quad \sum_{i=1}^n a_{ij} x_i \leq b_j, \quad j = 1, \dots, m \\ \text{et} \quad x_i \geq 0, \quad i = 1, \dots, n \end{array} \right.$$

Remarques

- Minimiser $f \iff$ Maximiser $-f$
- Les inégalités " \geq " se transforment en inégalités " \leq " en les multipliant par -1
- Une égalité " $=$ " revient à deux inégalités : " \geq " et " \leq "
- Si x est une variable négative, alors on définit $y = -x$ comme variable positive.
- Si x est une variable non-contrainte en signe, alors on définit deux nouvelles variables tel que $x = x^+ - x^-$ avec $x^+ \geq 0$ et $x^- \geq 0$
- Pas de contraintes d'inégalité stricte



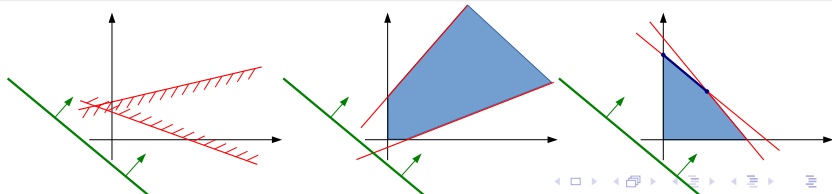
Définition : polyèdre des contraintes

- Ensemble admissible : ensemble des points de \mathbb{R}^n vérifiant les $m + n$ contraintes de (PL)
- ⇒ c'est un **polyèdre**
- ⇒ ce polyèdre est convexe
- Si un polyèdre est borné (distance des points à l'origine borné), alors on le nomme **polytope**

$$P = \{x \in \mathbb{R}^n : \sum_{i=1}^n a_{ij}x_i \leq b_j, j = 1, \dots, m; x_i \geq 0, i = 1, \dots, n\}$$

Théorème

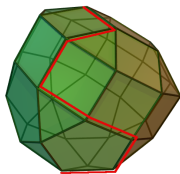
Le maximum d'un (PL) , s'il existe, est atteint au moins en un sommet du polyèdre



(PL) est un problème d'optimisation polynomial.

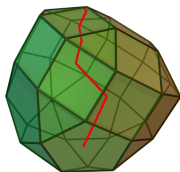
Il se résout souvent en pratique avec l'algorithme du simplexe de complexité exponentielle en théorie mais efficace en pratique.

Illustration de l'algorithme du simplexe sur un polyèdre 3D :



Il existe des algorithmes polynomiaux (algorithmes de points intérieurs).

Illustration d'un algorithme de points intérieurs sur un polyèdre 3D :



Variante 1 : programmation linéaire à variables entières

$$(PLNE) \left\{ \begin{array}{l} \text{Maximiser} \quad \sum_{i=1}^n c_i x_i \\ \text{sous les contraintes} \quad \sum_{i=1}^n a_{ij} x_i \leq b_j, \quad j = 1, \dots, m \\ \text{et} \quad x_i \in \mathbb{N}, \quad i = 1, \dots, n \end{array} \right.$$

Variante 2 : programmation linéaire à variables binaires

$$(PLVB) \left\{ \begin{array}{l} \text{Maximiser} \quad \sum_{i=1}^n c_i x_i \\ \text{sous les contraintes} \quad \sum_{i=1}^n a_{ij} x_i \leq b_j, \quad j = 1, \dots, m \\ \text{et} \quad x_i \in \{0, 1\}, \quad i = 1, \dots, n \end{array} \right.$$

Variante 3 : programmation linéaire à variables mixtes

$$(PLVM) \left\{ \begin{array}{l} \text{Maximiser} \quad \sum_{i=1}^n c_i x_i \\ \text{sous les contraintes} \quad \sum_{i=1}^n a_{ij} x_i \leq b_j, \quad j = 1, \dots, m \\ \text{et} \quad x_i \in \mathbb{N}, \quad i = 1, \dots, q \\ \text{et} \quad x_i \geq 0, \quad i = q + 1, \dots, n \end{array} \right.$$

(PLNE), (PLVB) et (PLVM) sont des problèmes d'optimisation NP-difficile dans le cas général. On les résout en général par des méthodes exactes de recherche arborescente (les méthodes de séparation et d'évaluation) ou bien par des méthodes approchées (diverses heuristiques).

On considère le polyèdre (P) décrit par les inégalités d'un PLVM et en remplaçant toutes les contraintes d'intégrité par des contraintes de positivité.

$$P = \{x \in \mathbb{R}^n : \sum_{i=1}^n a_{ij}x_i \leq b_j \text{ pour } j = 1, \dots, m, x_i \geq 0 \text{ pour } i = 1, \dots, n\}$$

Notons que l'ensemble des solutions réalisable du PLVM est l'ensemble des points entiers de P noté P^E :

$$P^E = P \cap (\mathbb{N}^q \times \mathbb{R}^{n-q})$$

On définit la relaxation de programmation linéaire d'un PLVM comme le PL suivant :

Maximiser $\sum_{i=1}^n c_i x_i$ sous les contraintes $x \in P$

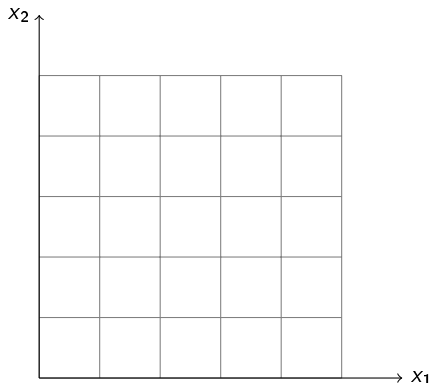
Théorème

Soit z^* la valeur optimale du PLVM, si elle existe, et \tilde{z}^* la valeur optimale de sa relaxation de PL, si elle existe. On a $\tilde{z}^* \geq z^*$.

- Problèmes d'optimisation combinatoire
 - L'ensemble des solutions admissibles n'est plus continu.
 - On ne peut plus compter sur les sommets du polyèdre.
 - On ne peut généralement pas énumérer toutes les solutions.
 - On ne peut pas se contenter de la solution entière la plus proche de l'optimum continu.

Exemple :

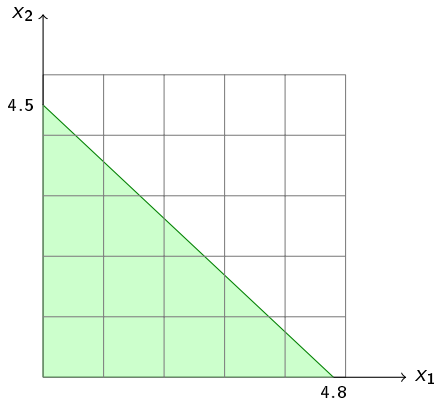
Maximiser $z = 19x_1 + 20x_2$
Sous les contraintes
 $15x_1 + 16x_2 \leq 72$
 $x_1, x_2 \in \mathbb{N}$



- Problèmes d'optimisation combinatoire
 - L'ensemble des solutions admissibles n'est plus continu.
 - On ne peut plus compter sur les sommets du polyèdre.
 - On ne peut généralement pas énumérer toutes les solutions.
 - On ne peut pas se contenter de la solution entière la plus proche de l'optimum continu.

Exemple :

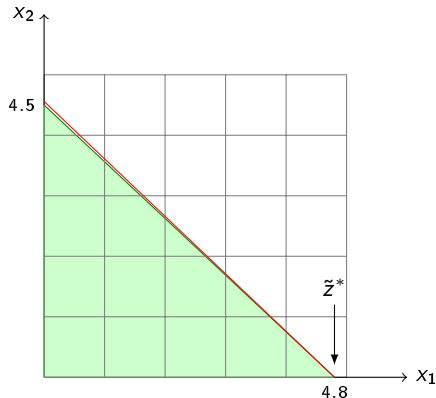
Maximiser $\tilde{z} = 19x_1 + 20x_2$
Sous les contraintes
 $15x_1 + 16x_2 \leq 72$
 $x_1, x_2 \in \mathbb{R}^+$



- Problèmes d'optimisation combinatoire
 - L'ensemble des solutions admissibles n'est plus continu.
 - On ne peut plus compter sur les sommets du polyèdre.
 - On ne peut généralement pas énumérer toutes les solutions.
 - On ne peut pas se contenter de la solution entière la plus proche de l'optimum continu.

Exemple :

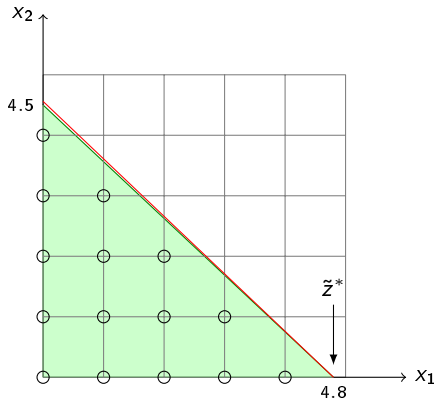
Maximiser $\tilde{z} = 19x_1 + 20x_2$
Sous les contraintes
 $15x_1 + 16x_2 \leq 72$
 $x_1, x_2 \in \mathbb{R}^+$



- Problèmes d'optimisation combinatoire
 - L'ensemble des solutions admissibles n'est plus continu.
 - On ne peut plus compter sur les sommets du polyèdre.
 - On ne peut généralement pas énumérer toutes les solutions.
 - On ne peut pas se contenter de la solution entière la plus proche de l'optimum continu.

Exemple :

Maximiser $z = 19x_1 + 20x_2$
Sous les contraintes
 $15x_1 + 16x_2 \leq 72$
 $x_1, x_2 \in \mathbb{N}$

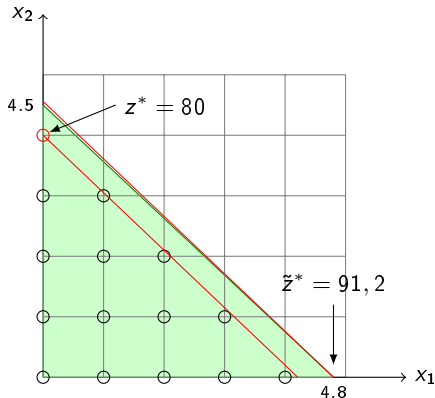


- Problèmes d'optimisation combinatoire

- L'ensemble des solutions admissibles n'est plus continu.
- On ne peut plus compter sur les sommets du polyèdre.
- On ne peut généralement pas énumérer toutes les solutions.
- On ne peut pas se contenter de la solution entière la plus proche de l'optimum continu.

Exemple :

Maximiser $z = 19x_1 + 20x_2$
Sous les contraintes
 $15x_1 + 16x_2 \leq 72$
 $x_1, x_2 \in \mathbb{N}$



L'intérêt des programmes linéaires (en nombres entiers) est qu'il existe de nombreux solveurs disponible sur le marché (solveurs commerciaux) ou bien en libre accès. Ces solveurs implémentent les algorithmes standards, plus ou moins performants, pour résoudre les PL ou les PLNE.

Quelques solveurs de PLNE :

- IBM CPLEX
- GUROBI
- SCIP
- FICO XPRESS
- COIN-OR CBC / CLP
- LINGO
- MINTO
- LPSOLVE
- GLPK
- ...

A côté des solveurs, on trouve des langages de modélisation, permettant d'exprimer de manière algébrique des problèmes d'optimisation (linéaires ou non) comme AIMMS, AMPL, LINDO, MPL ou OPL, ...

Soit un problème d'optimisation général (PO) comportant $p \geq 0$ variables entières et $q \geq 0$ variables continues :

(PO) Maximiser $f(x)$ sous les contraintes $x \in \mathcal{X} \subset \mathbb{N}^p \times \mathbb{R}^q$

f et les contraintes définissant \mathcal{X} ne sont pas nécessairement linéaires.

La modélisation consiste à trouver une bonne formulation (PL) ou (PLNE) de (PO).

- Une formulation (PL) compacte (avec un nombre polynomial de variables et de contraintes) ne peut être trouvée que pour les problèmes polynomiaux¹. L'objectif est alors de résoudre le problème avec l'algorithme du simplexe (ou autre algorithme de PL), implémenté dans les solveurs sur étagère.
- On utilise la PLNE pour modéliser les problèmes NP-difficiles. L'objectif est alors de résoudre le problème au moyen des algorithmes de séparation/évaluation et/ou de coupes, également implémentés dans les solveurs sur étagère.

1. sauf si $P=NP$

Formulation PLVM (cas général)

Dans le cas le plus général, cette formulation s'écrit comme un PLVM en ajoutant $p' \geq 0$ variables entières $y_1, \dots, y_{p'}$ et/ou $q' \geq 0$ variables continues $y_{p'+1}, \dots, y_{p'+q'}$ nécessaires à la linéarisation.

$$(PLVM) \left\{ \begin{array}{ll} \text{Maximiser} & \sum_{i=1}^{p+q} c_i x_i + \sum_{i=1}^{p'+q'} c'_i y_i \\ \text{sous les contraintes} & \sum_{i=1}^{p+q} a_{ij} x_i + \sum_{i=1}^{p'+q'} a'_{ij} y_i \leq b_j, \quad j = 1, \dots, m \\ \text{et} & x_i \in \mathbb{N}, \quad i = 1, \dots, p \\ \text{et} & x_i \geq 0, \quad i = p+1, \dots, p+q \\ \text{et} & y_i \in \mathbb{N}, \quad i = 1, \dots, p' \\ \text{et} & y_i \geq 0, \quad i = p'+1, \dots, p'+q' \end{array} \right.$$

Soit P le polyèdre défini par le PLVM.

Soit $P^E = P \cap \mathbb{N}^{p+p'} \times \mathbb{R}^{q+q'}$ l'ensemble des solutions du PLVM.

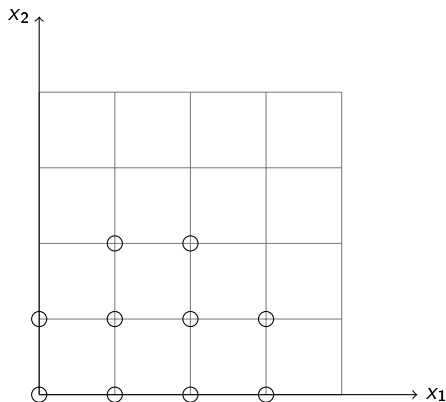
Soit $proj_x(P^E)$ la projection de P^E sur l'espace des variables originales :

$$proj_x(P^E) = \left\{ x \in \mathbb{N}^p \times \mathbb{R}^q : (x, y) \in P^E, \text{ pour un certain } y \in \mathbb{N}^{p'} \times \mathbb{R}^{q'} \right\}$$

La formulation est correcte si

- $\mathcal{X} = proj_x P^E$
- et si pour tout $x \in \mathcal{X}$, $f(x) = c^T x + c'^T y^*$ où y^* est la solution optimale du PLVM en ayant fixé les variables x .

Soit l'ensemble \mathcal{X} constitué des points entiers ci-dessous et une fonction linéaire $f(x_1, x_2)$ à maximiser. On cherche une formulation correcte pour tout f .



Soit l'ensemble X constitué des points entiers ci-dessous et une fonction linéaire $f(x_1, x_2)$ à maximiser. On cherche une formulation correcte pour tout f .

Formulation correcte 1

Maximiser $z = f(x_1, x_2)$

Sous les contraintes

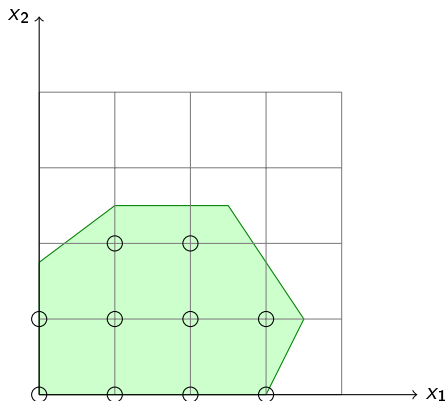
$$-3x_1 + 4x_2 \leq 7$$

$$2x_2 \leq 5$$

$$6x_1 + 4x_2 \leq 25$$

$$2x_1 - x_2 \leq 6$$

$$x_1, x_2 \in \mathbb{N}$$



Soit une fonction f linéaire $f(x) = c^T x$ et le problème d'optimisation

$$(PO) \text{ Maximiser } f(x) \text{ sous les contraintes } x \in \mathcal{X} \text{ avec } \mathcal{X} \subset \mathbb{R}^n$$

On définit l'enveloppe convexe de \mathcal{X} , notée $\text{conv}(\mathcal{X})$ comme :

$$\text{conv}(\mathcal{X}) = \{x : x = \sum_{i=1}^t \lambda_i x^i, \sum_{i=1}^t \lambda_i = 1, \lambda_i \geq 0 \text{ pour } i = 1, \dots, t \text{ sur tous les sous-ensembles finis } \{x^1, \dots, x^t\} \text{ de } \mathcal{X}\}$$

Théorème

$\text{conv}(\mathcal{X})$ est un polyèdre et (PO) est équivalent au PL

$$\text{Maximiser } c^T x \text{ sous les contraintes } x \in \text{conv}(\mathcal{X})$$

Il existe donc –en théorie– une formulation PL en les variables originales dont la résolution donne l'optimum de (PO).

Mais en pratique cette formulation nécessite un nombre exponentiel d'inégalités, par ailleurs difficiles à caractériser.

Soit P le polyèdre d'un PLVM définissant une formulation correcte de (PO).

Soit $proj_x(P)$ la projection de P sur l'espace des variables originales.

Le PLVM est une formulation idéale de PO si

$$proj_x(P) = conv(\mathcal{X})$$

En effet dans ce cas la valeur optimale du PL est $\tilde{z}^*(P) = z^*$ (l'optimum de PO coïncide avec l'optimum continu de la relaxation du PLVM).

Une formulation définissant un polyèdre P est meilleure qu'une formulation définissant un polyèdre P' si

$$proj_x(P) \subset proj_x(P')$$

En effet, dans ce cas, $z^* \leq \tilde{z}^*(P) \leq \tilde{z}^*(P')$. P définit une meilleure relaxation PL que P' .

Formulation correcte 2
Maximiser $z = f(x_1, x_2)$
Sous les contraintes

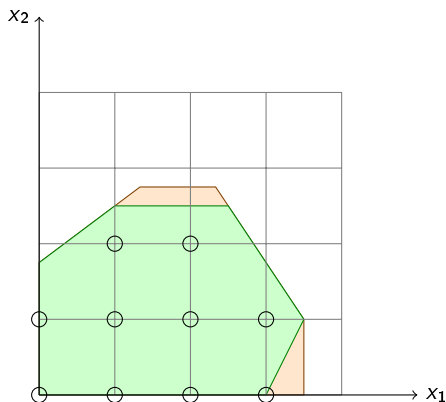
$$-3x_1 + 4x_2 \leq 7$$

$$4x_2 \leq 11$$

$$6x_1 + 4x_2 \leq 25$$

$$2x_1 \leq 7$$

$$x_1, x_2 \in \mathbb{N}$$



Formulation correcte 3
Maximiser $z = f(x_1, x_2)$
Sous les contraintes

$$-x_1 + x_2 \leq 1$$

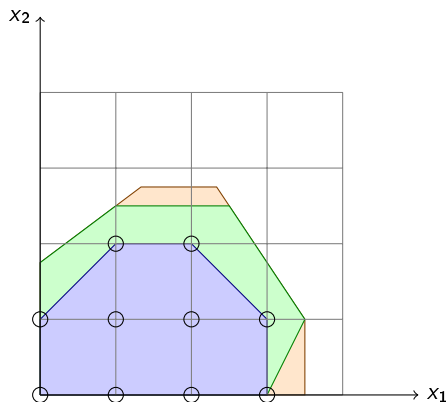
$$x_2 \leq 2$$

$$x_1 + x_2 \leq 4$$

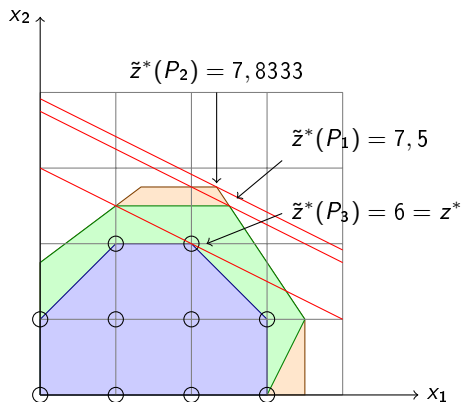
$$x_1 \leq 3$$

$$x_1, x_2 \in \mathbb{N}$$

(formulation idéale)

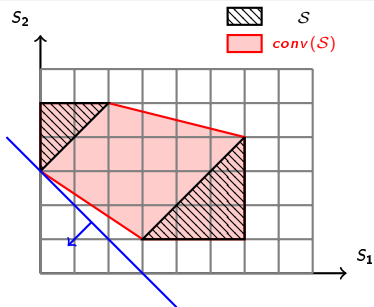


exemple pour $f = x_1 + 2x_2$



Exemple

Un simple problème d'ordonnancement à deux tâches J_1 et J_2 de durées connues $p_1 = 3$ et $p_2 = 2$ sur une machine ne pouvant exécuter qu'une tâche à la fois. On doit déterminer les dates de débuts continues S_1 et S_2 sachant que chaque tâche doit démarrer dans une fenêtre de temps ($[0, 6]$ pour J_1 et $[1, 5]$ pour J_2). On cherche à minimiser $f(S) = S_1 + S_2 + p_1 + p_2$.



(PO) peut être résolu par la PL sur $conv(S)$.
 Impraticable en général

(PO) Minimiser $S_1 + S_2 + 5$

$$S_1 \geq 0$$

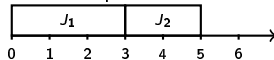
$$S_2 \geq 1$$

$$S_1 \leq 6$$

$$S_2 \leq 5$$

$$S_2 \geq S_1 + 3 \vee S_1 \geq S_2 + 2$$

Solution optimale de valeur 8



- On introduit une variable binaire pour représenter les deux alternatives

(PLVM) Minimiser $S_1 + S_2 + 5$

$$S_1 \geq 0$$

$$S_2 \geq 1$$

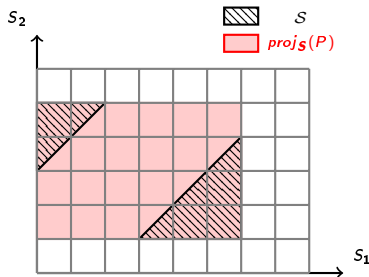
$$S_1 \leq 6$$

$$S_2 \leq 5$$

$$S_2 - S_1 + 8x \geq 3$$

$$S_1 - S_2 + 7(1 - x) \geq 2$$

$$x \in \{0, 1\}$$



La projection de P^E sur \mathbb{R}^2 donne bien S

Valeur de relaxation = 6

Problème : $x = 0.5$ réalisable

$projs(P)$ est éloigné de $conv(S)$

Problème de localisation d'entrepôt

Une entreprise livre des produits à m clients et peut construire pour cela n entrepôts potentiels. Ouvrir un entrepôt j entraîne un coût fixe f_j et le transport d'une commande de l'entrepôt i vers le client j a également un coût c_{ij} de telle sorte que si on transporte une fraction de la commande d'un client j depuis l'entrepôt i , le coût est de c_{ij} fois cette fraction. Quels entrepôts doivent être ouverts et quel est le plan de transport pour couvrir 100% des commandes à cout minimal ?

- Ouverture d'un entrepôt : variable binaire y_i
- Pourcentage de la commande du client i transportée depuis l'entrepôt j : variable continue $x_{ij} \in [0, 1]$.

$$\text{Minimiser } \sum_{i=1}^n \sum_{j=1}^m c_{ij} x_{ij} + \sum_{i=1}^n p_i y_i$$

$$\text{Sous les contraintes } \left\{ \begin{array}{ll} \sum_{i=1}^n x_{ij} = 1 & \forall j = 1, \dots, m \\ \sum_{j=1}^m x_{ij} \leq m y_i & \forall i = 1, \dots, n \\ y_i \in \{0, 1\} & \forall i = 1, \dots, n \\ x_{ij} \geq 0 & \forall i = 1, \dots, n; \forall j = 1, \dots, m \end{array} \right.$$

La contrainte $\sum_{j=1}^m x_{ij} \leq my_i, \forall i = 1, \dots, n$ (1) modélise l'implication

$$y_i = 0 \implies \sum_{j=1}^m x_{ij} = 0$$

Une alternative est de modéliser m implications

$$y_i = 0 \implies x_{ij} = 0, \forall j = 1, \dots, m$$

On obtient des contraintes désagrégées

$$x_{ij} \leq y_i, \forall i = 1, \dots, n, \forall j = 1, \dots, m$$
 (2)

Théorème

La formulation désagrégée est strictement meilleure que la formulation agrégée.

Preuve. Soit P_1 le polyèdre correspondant aux contraintes agrégées (1) et soit P_2 le polyèdre remplaçant les contraintes (1) par les contraintes désagrégées (2). Comme (2) \implies (1), on a $P_2 \subseteq P_1$. Pour prouver l'inclusion stricte il suffit de trouver un point $x \in P_1 \setminus P_2$. Supposons que $m = kn$. Soit le point x tel que $x_{ij} = 1$ pour $i = 1, \dots, n$; $j = k(j-1) + 1, \dots, k(j-1) + k$; $x_{ij} = 0$ sinon et $y_i = k/m$ pour $i = 1, \dots, n$. x ainsi défini satisfait (1) mais viole (2).

Classe de PLNE tels que, quelle que soit la valeur de b , la relaxation PL donne toujours une solution entière.

Propriété

Soit $B \in \mathbb{Z}^{m \times m}$ une matrice non singulière, alors $B^{-1}b$ est entier pour tout $b \in \mathbb{Z}^m$ si et seulement si $\text{Det}(B) = +\text{ ou } -1$.

Une matrice $A \in \mathbb{Z}^{m \times p}$ de plein rang est **unimodulaire** si $\text{Det}(B) = +\text{ ou } -1$ pour toute base B de A .

Une matrice est **totalemment unimodulaire** si chacune de ses sous-matrices a un déterminant égal à 0, 1 ou -1.

Corollaire

Si A est totalement unimodulaire, il existe une solution optimale de la relaxation continue qui résout le problème en nombres entiers (la formulation est idéale).

Conditions suffisantes d'unimodularité :

- $a_{ij} \in \{-1; 0; 1\} \quad \forall i, j$
- $\sum_{i=1}^m |a_{ij}| \leq 2 \quad \forall j$
- Il existe une partition M_1, M_2 des indices des lignes telle que, si $\sum_{i=1}^m |a_{ij}| = 2$, alors $\sum_{i \in M_1} a_{ij} = \sum_{i \in M_2} a_{ij}$.

Problème d'affectation

n tâches doivent être affectées à n employés. Comme certains employés sont plus expérimentés que d'autres pour effectuer certaines tâches, c_{ij} représente le temps mis par un employé i pour effectuer la tâche j . Quel est l'affectation qui minimise le temps de travail total ?

Modélisation : variable $x_{i,j} \in \{0, 1\}$ qui vaut 1 si l'employé i est affecté à la tâche j .

$$\begin{array}{l} \text{Minimiser } \sum_{i=1}^n \sum_{j=1}^n c_{ij} x_{ij} \\ \text{sous les contraintes } \left\{ \begin{array}{ll} \sum_{j=1}^n x_{ij} = 1 & \forall j = 1, \dots, n \\ \sum_{i=1}^n x_{ij} = 1 & \forall i = 1, \dots, n \\ x_{ij} \in \{0, 1\} & \forall i = 1, \dots, n; \forall j = 1, \dots, n \end{array} \right. \end{array}$$

Exemple de matrice A pour $n = 3$:

1	0	0	1	0	0	1	0	0
0	1	0	0	1	0	0	1	0
0	0	1	0	0	1	0	0	1
1	1	1	0	0	0	0	0	0
0	0	0	1	1	1	0	0	0
0	0	0	0	0	0	1	1	1

A vérifie les conditions suffisantes de TU avec $M1 = \{1, 2, 3\}$ et $M2 = \{4, 5, 6\}$

On peut remplacer la contrainte

$x_{ij} \in \{0, 1\}; i = 1, \dots, n; j = 1, \dots, n$ par $x_{ij} \geq 0; i = 1, \dots, n; j = 1, \dots, n$. Le problème est donc polynomial.

Certaines fonctions non linéaires peuvent être systématiquement formulées de manière linéaire en ajoutant des variables entières.

- $f_1 : \{0, 1\}^2 \mapsto \{0, 1\}, z = f_1(x, y) = xy$

$$\begin{cases} z \leq x \\ z \leq y \\ z \geq x + y - 1 \\ x, y \in \{0, 1\} \end{cases}$$

- $f_2 : \{0, 1\} \times [0, a] \mapsto \mathbb{R}, z = f_2(x, y) = xy, \text{ avec } a \in \mathbb{R}$

$$\begin{cases} z \leq ax \\ z \leq y \\ z \geq y - (1 - x)a \\ x \in \{0, 1\} \\ 0 \leq y \leq a \end{cases}$$

- $f_3 : [0, C]^2 \mapsto [0, C], z = f_3(x, y) = \text{Minimiser}(x, y)$

$$\left\{ \begin{array}{l} z \leq x \\ z \leq y \\ x \leq y + Cw \\ y \leq x + C(1 - w) \\ z \geq (1 - w)x + wy \\ w \in \{0, 1\} \\ 0 \leq x \leq C \\ 0 \leq y \leq C \end{array} \right.$$

La modélisation de f_3 peut être simplifiée si le coefficient de z dans l'objectif est positif, dans le cadre d'une maximisation

Maximiser az

Sous les contraintes $\left\{ \begin{array}{l} z \leq x \\ z \leq y \\ 0 \leq x \leq C \\ 0 \leq y \leq C \end{array} \right.$

avec $a > 0$

- $f_4 : [a, b] \mapsto \mathbb{R}, z = f_4(x) = |x|$ avec $a < 0 < b$

$$\left\{ \begin{array}{l} z = x^+ + x^- \\ x = x^+ - x^- \\ 0 \leq x^+ \leq by \\ 0 \leq x^- \leq |a|(1 - y) \\ a \leq x \leq b \\ y \in \{0, 1\} \end{array} \right.$$

Problème d'affectation

n tâches doivent être affectées à n employés. Comme certains employés sont plus expérimentés que d'autres pour effectuer certaines tâches, c_{ij} représente le temps mis par un employé i pour effectuer la tâche j . Quel est l'affectation qui minimise le temps de travail total ?

Application spatiale : affectation de plages de temps dans les systèmes de télécommunications par satellite TDMA [Bur85]

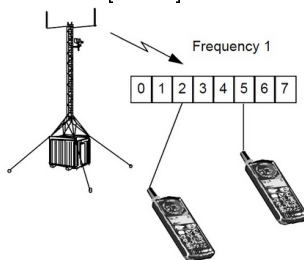
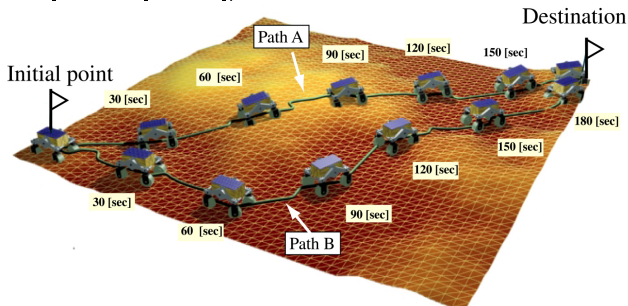


Figure TDMA

Plus court chemin

Un réseau routier est représenté par un graphe orienté $G = (V, A)$. Le transport direct d'un nœud i à un nœud j du réseau est possible si $(i, j) \in A$ and a un coût c_{ij} . Etant donné deux nœuds s et t , quel est le chemin de coût minimum de s à t ?

Application spatiale : Navigation autonome d'un rover sur Mars (Canadian Space Agency : [DRB⁺08], LAAS [HST02])



Formulation du problème du plus court chemin

On utilise la variable binaire $x_{ij} = 1$ si le plus court chemin passe par l'arc $(i, j) \in A$.

Soit $V^+(i)$ l'ensemble des successeurs immédiats de i et soit $V^-(i)$ l'ensemble des prédécesseurs immédiats de i .

Minimiser $\sum_{(i,j) \in A} c_{ij} x_{ij}$

Sous les contraintes

$$\begin{cases} \sum_{k \in V^+(s)} x_{sk} - \sum_{k \in V^-(s)} x_{ks} = 1 \\ \sum_{k \in V^+(i)} x_{ik} - \sum_{k \in V^-(i)} x_{ki} = 0 \\ \sum_{k \in V^+(t)} x_{tk} - \sum_{k \in V^-(s)} x_{kt} = -1 \\ x_{ij} \in \{0, 1\} \end{cases} \quad \begin{array}{l} \forall i \in V \setminus \{s, t\} \\ \\ \\ \forall (i, j) \in A \end{array}$$

On observe encore que la matrice des contraintes est TU par application des conditions suffisantes.

On peut remplacer la contrainte $x_{ij} \in \{0, 1\}, \forall (i, j) \in A$ par $x_{ij} \geq 0, \forall (i, j) \in A$.

Le problème est polynomial.

Comme on obtient un PL, on peut utiliser la forme duale. Soit π_i la variable duale de la contrainte liée au sommet i de V . On obtient :

Maximiser $\pi_t - \pi_s$

Sous les contraintes

$$\begin{cases} \pi_j - \pi_i \leq c_{ij} & \forall (i, j) \in A \\ \pi_i \geq 0 & \forall i \in V \end{cases}$$

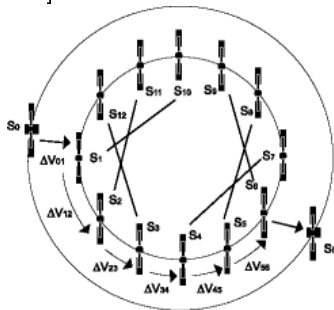
On peut poser $\pi_s = 0$ sans perte de généralité.

La recherche du plus court chemin est donc équivalente à la recherche de potentiels π_i pour chaque sommet tels que l'inégalité de potentiels $\pi_j - \pi_i \leq c_{ij}$ est vérifiée pour chaque arc (propriété à la base des algorithmes de plus court chemin dédiés)

Problème de flot à coût minimum

Un réseau de transport est représenté par un graphe orienté $G = (V, A)$. Chaque nœud $i \in V$ a une valeur b_i , correspondant à une quantité produite ($b_i > 0$), une demande ($b_i < 0$) ou un transit ($b_i = 0$) de produits. Chaque arc $(i, j) \in A$ représente une ressource de transport avec une capacité limitée h_{ij} et un coût unitaire c_{ij} . Quel est le plan de transport satisfaisant toutes les demandes à coût minimum ?

Application spatiale : Planification des manœuvres de de réapprovisionnement des satellites en constellation [Dut09]



On utilise la variable continue x_{ij} donnant la quantité de produit transitant par l'arc $(i, j) \in A$.

Minimiser $\sum_{(i,j) \in A} c_{ij} x_{ij}$

Sous les contraintes

$$\begin{cases} \sum_{k \in V^+(i)} x_{ik} - \sum_{k \in V^-(i)} x_{ki} = b_i & \forall i \in V \\ 0 \leq x_{ij} \leq h_{ij} & \forall (i, j) \in A \end{cases}$$

On peut également montrer que la matrice des contraintes est TU. Donc si les demandes et les capacités sont entières, les valeurs x_{ij} seront entières. Le problème reste donc polynomial si on impose des flots entiers.

Le problème de plus court chemin et le problème d'affectation sont des cas particulier du problème de flot à coût minimal.

Problème du sac à dos

Un sac à dos a une capacité de B kg. On doit choisir quels objets emporter dans un ensemble de n objets sachant que chaque objet possède un poids c_i et un profit v_i . L'objectif est d'emporter l'ensemble d'objets de profit maximum en respectant la capacité du sac à dos.

Application spatiale : sélection d'un ensemble de prises de vues par un satellite d'observation avec une capacité d'enregistrement à bord B . Poids = taille de la prise de vue, Profit = gain commercial si la prise de vue est effectuée. Problème fourni par le CNES (exemple : satellite SPOT 5) [VH01]



On utilise une variable binaire $x_i = 1$ si l'objet i est emporté.

Maximiser $\sum_{i=1}^n v_i x_i$

Sous les contraintes

$$\begin{cases} \sum_{i=1}^n c_i x_i \leq B \\ x_i \in \{0, 1\} \end{cases} \quad \forall i = 1, \dots, n$$

Le problème est NP-difficile au sens ordinaire.

Problème de couverture par ensembles

Un ensemble de m défauts est à identifier par un ensemble de n tests sachant que chaque test i se déclenche sur (ou couvre) un ensemble de S_i défauts. La mise en place d'un test a un coût c_i . Quels tests doivent être sélectionnés pour couvrir l'ensemble des défauts à coût minimum ?

Application spatiale : Diagnostic embarqué dans les stations spatiales (système TROUBLE 3 de la NASA) [Man90]

Mars Curiosity

March 3, 2015

Testing to Diagnose Power Event in Mars Rover



On utilise une variable binaire $x_i = 1$ si le test i est mis en place.

On utilise le paramètre a_{ij} qui vaut 1 si le test i se déclenche sur le défaut j ($j \in S_i$).

Minimiser $\sum_{i=1}^n c_i x_i$

Sous les contraintes

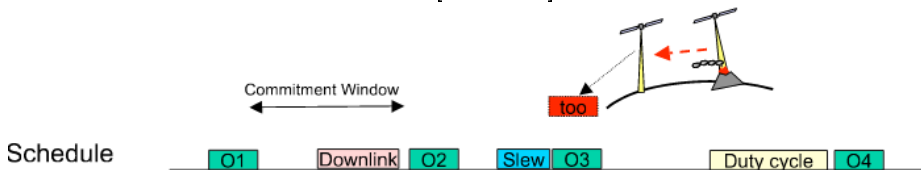
$$\begin{cases} \sum_{i=1}^n a_{ij} x_i \geq 1 & \forall j = 1, \dots, m \\ x_i \in \{0, 1\} & \forall i = 1, \dots, n \end{cases}$$

Le problème est NP-difficile au sens fort.

Ordonnancement à une machine

Une machine doit fabriquer n produits. Chaque produit i a une durée de fabrication p_i , les composants nécessaires à la fabrication du produit arrivent à la date de lancement r_i et une date de livraison d_i est négociée avec le client. Sachant que la machine ne peut fabriquer qu'un produit à la fois et que la fabrication d'un produit commencé ne peut pas être interrompue, quel est l'ordonnancement des tâches qui minimise le nombre de tâches exécutées après leur date annoncée de livraison ?

Application spatiale : ordonnancement des communications bord/sol dans le réseaux de contrôle des satellites de l'U.S. Air Force [BWWH04]



Formulation du problème d'ordonnancement à une machine

On propose un PLVM inspiré du modèle "disjonctif". On utilise une variable binaire $x_{ij} = 1$ si la tâche i est ordonnancée avant la tâche j sur la machine et 0 sinon.

On utilise également une variable continue S_i de date de début de chaque tâche i

Enfin, on définit une variable binaire $y_i = 1$ si la tâche i est en retard.

Minimiser $\sum_{i=1}^n y_i$

Sous les contraintes

$$\left\{ \begin{array}{ll} S_j \geq S_i + p_i - M(1 - x_{ij}) & \forall i, j \in \{1, \dots, n\}, i \neq j \\ S_i \leq d_i + My_i & \forall i = 1, \dots, n \\ x_{ij} + x_{ji} = 1 & \forall i, j \in \{1, \dots, n\}, i \neq j \\ x_{ij} \in \{0, 1\} & \forall i, j \in \{1, \dots, n\}, i \neq j \\ y_i \in \{0, 1\} & \forall i = 1, \dots, n \\ S_i \geq r_i & \forall i = 1, \dots, n \end{array} \right.$$

M est une constante suffisamment grande pour rendre les contraintes vérifiées si la variable binaire dont elle est le coefficient vaut 1. On peut prendre $M = \max_{i=1, \dots, n} r_i + \sum_{i=1}^n p_i$.

Le problème est NP-difficile au sens fort.

Problème du voyageur de commerce

Un voyageur de commerce doit visiter n villes en commençant par la ville 1 et en revenant à cette même ville. c_{ij} est le coût de trajet de la ville i à la ville j . Quelle est la tournée de coût minimum ?

Application spatiale : problème de collecte de débris spatiaux : collecter un ensemble de débris spatiaux en minimisant la consommation de carburant.



Proposition d'une mission de collecte de débris spatiaux dans l'orbite LEO (EADS-ASTRIUM) [Cer11]

Formulation du problème de voyageur de commerce

On définit une variable binaire $x_{ij} = 1$ si le voyageur va directement de la ville i à la ville j avec $x_{ji} := 0$

Minimiser $\sum_{i=1}^n \sum_{j=1}^n c_{ij} x_{ij}$

Sous les contraintes

$$\begin{cases} \sum_{j \in \{1, \dots, n\} \setminus \{i\}} x_{ij} = 1 & \forall i = 1, \dots, n \\ \sum_{i \in \{1, \dots, n\} \setminus \{j\}} x_{ij} = 1 & \forall j = 1, \dots, n \\ \sum_{i \in S} \sum_{j \in \{1, \dots, n\} \setminus S} x_{ij} \geq 1 & \forall S \subset \{1, \dots, n\}, S \neq \emptyset \\ x_{ij} \in \{0, 1\} & \forall i, j \in \{1, \dots, n\}, i \neq j \end{cases}$$

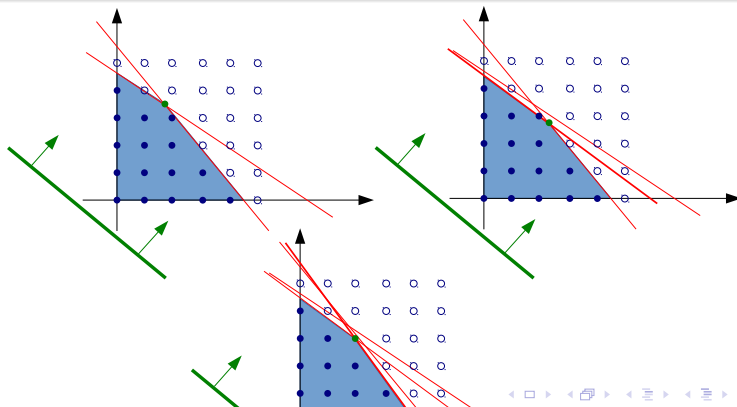
Les contraintes sont nécessaires et suffisantes pour que les variables x_{ij} définissent un tour complet sur toutes les villes :

- Contraintes 1 : Toute ville est suivie d'une autre ville dans le tour
- Contraintes 2 : Toute ville est précédée d'une autre ville dans le tour
- Contraintes 3 : Les deux premières contraintes n'empêchent pas la formation de sous-tours (Exemple pour $n = 5$: un sous-tour $1 \rightarrow 2 \rightarrow 3 \rightarrow 1$ et un autre sous-tour déconnecté $4 \rightarrow 5 \rightarrow 4$). On pose alors que, pour tout ensemble de sommet S susceptible de former un sous-tour, le tour doit quitter cet ensemble par au moins un arc. Inconvénient : il y a 2^n contraintes de ce type !

Le problème est NP-difficile au sens fort. Nous verrons plus loin comment gérer l'explosion combinatoire des contraintes de sous-tour.

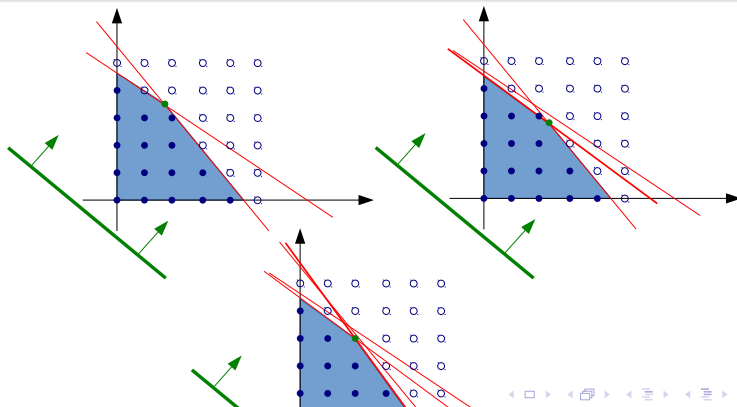
Principe

- 1 Trouver la solution continue du PLNE par la méthode du simplexe.



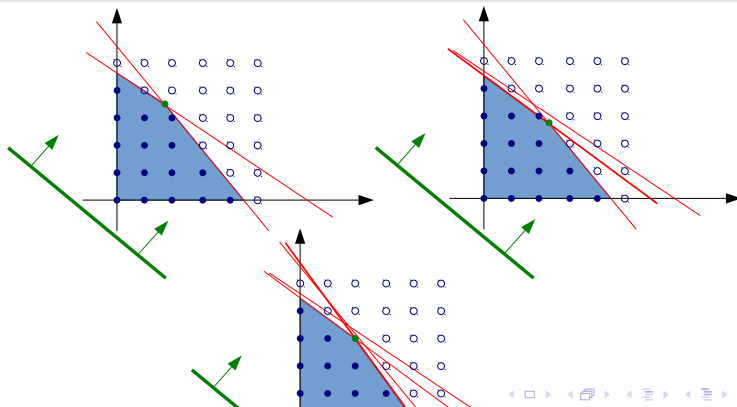
Principe

- 1 Trouver la solution continue du PLNE par la méthode du simplexe.
- 2 Ajouter une contrainte (appelée coupe) qui :



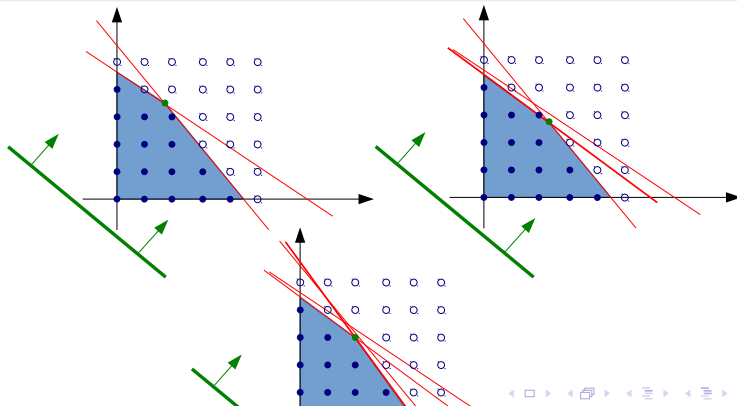
Principe

- 1 Trouver la solution continue du PLNE par la méthode du simplexe.
- 2 Ajouter une contrainte (appelée coupe) qui :
 - 1 supprime la solution continue optimale et



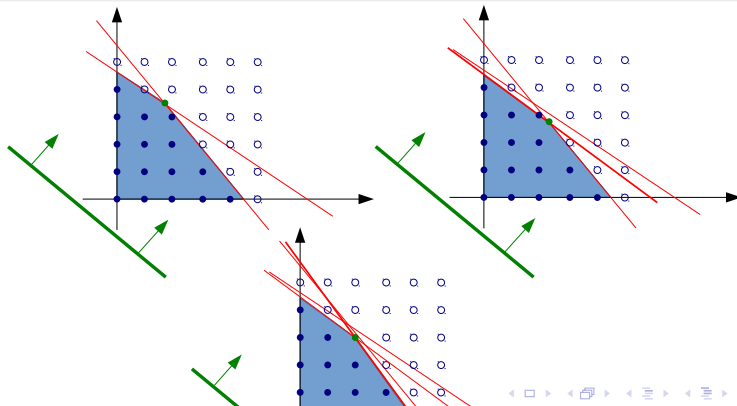
Principe

- 1 Trouver la solution continue du PLNE par la méthode du simplexe.
- 2 Ajouter une contrainte (appelée coupe) qui :
 - 1 supprime la solution continue optimale et
 - 2 garde toutes les solutions entières.



Principe

- 1 Trouver la solution continue du PLNE par la méthode du simplexe.
- 2 Ajouter une contrainte (appelée coupe) qui :
 - 1 supprime la solution continue optimale et
 - 2 garde toutes les solutions entières.
- 3 Itérer jusqu'à trouver une solution continue entière.



Comment trouver une coupe à ajouter, violée par la solution optimale continue ?

- Coupes génériques (ex : coupes de Gomory) : peuvent s'appliquer sur n'importe quel PLNE
- Coupes spécifiques : s'appliquent sur des problèmes particuliers

Etant donné une famille de coupe, le problème de séparation² consiste, à partir de la solution continue du problème relâché, à trouver une coupe violée par cette solution ou à prouver qu'il n'en existe pas.

- Dans le cas général, le problème de séparation est difficile, mais il existe certains cas polynomiaux.
- Les méthodes de coupe convergent en général très lentement et sont plutôt intégrées aux méthodes de séparation et d'évaluation pour améliorer la relaxation à chaque nœud de l'arbre de recherche.

2. Le *problème* de séparation (visant à chercher à savoir si une solution continue viole une certaine inégalité) est à distinguer de la *méthode* de séparation et d'évaluation qui cherche à séparer récursivement l'espace de recherche en plusieurs sous-parties pour résoudre le problème (voir plus loin)

Coupe générique : une recette des coupes de Gomory

Pour la résolution de la relaxation PL, l'algorithme du simplexe reformule le problème sous forme standard, en remplaçant toutes les inégalités par des égalités $Ax + e = b$ en introduisant au préalable des variables d'écart e .

Le système initial d'égalités est modifié par pivotage à chaque itération du simplexe, et l'optimum continu correspond à un système final d'égalités $A'x + De = b'$

La coupe de Gomory est basée sur l'analyse des parties fractionnaires des égalités. On suppose que toutes les variables sont entières.

Construction d'une coupe de Gomory pour une égalité

$$\sum_{i=1}^n a'_{ij}x_i + \sum_{i=1}^m d_{ij}e_j = b'_j \text{ donnée :}$$

Etape 1 : Exprimer chacun des coefficients de la contraintes comme la somme d'un entier et d'une fraction non négative (représentation unique)

Etape 2 Enlever les termes entiers et changer l'égalité en \geq

Exemple : égalité $0x_1 - 3/4x_2 + 1/3x_3 - 13/4x_4 = 73/7$

Etape 1 : $(0 + 0/1)x_1 + (-1 + 1/4)x_2 + (0 + 1/3)x_3 + (-4 + 3/4)x_4 = 10 + 3/7$

Etape 2 : $1/4x_2 + 1/3x_3 + 4/4x_4 \geq 3/7$

Pourquoi est-ce valide (voir

<http://www.ms.unimelb.edu.au/~moshe/620-362/gomory/>) ?

- **Problème du sac à dos.** Soit un ensemble d'objets C tels que $\sum_{i \in C} c_i > B$. C est appelé une couverture. La coupe de couverture est

$$\sum_{i \in C} x_i \leq |C| - 1$$

Il est intéressant de générer des coupes de couvertures minimales, c-à-d telles que si pour tout $C' \subset C$, C' n'est pas une couverture.

- **Problème de coloration de graphe.** Soit un graphe non orienté $G = (V, E)$. Etant donné un nombre de couleurs m , associer une couleur à chaque sommet de sorte que chaque sommet n'ait pas la même couleur que ses voisins.

$$\begin{cases} \sum_{k=1}^m x_{ik} = 1 & \forall i \in V \\ x_{ik} + x_{jk} \leq 1 & \forall (i, j) \in E; \forall k = 1, \dots, m \\ x_{ik} \in \{0, 1\} & \forall i \in V; \forall k = 1, \dots, m \end{cases}$$

Soit une clique du graphe G . La coupe de clique ci-dessous est valide.

$$\sum_{i \in C} x_{ij} \leq 1, \forall j = 1, \dots, m$$

Il est intéressant de générer des coupes de cliques maximales.

Etant donné une solution continue, trouver une inégalité valide de couverture ou de clique violées est NP-difficile.

Soit le problème du voyageur de commerce symétrique ($c_{ij} = c_{ji}$). On note $G(V, E)$ le graphe de la version symétrique et donc non orientée du PVC. La formulation se simplifie. On note x_e la variable binaire de sélection d'une arête dans le tour. On note $\delta(i)$ l'ensemble des arêtes incidentes au sommet i et $C(S)$ l'ensemble des arêtes reliant l'ensemble de sommets S aux autres sommets.

Minimiser $\sum_{e \in E} x_e$

Sous les contraintes

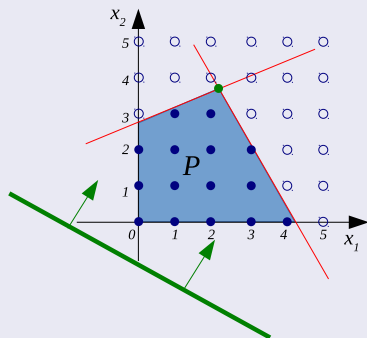
$$\begin{cases} \sum_{e \in \delta(i)} x_e = 2 & \forall i = 1, \dots, n \\ \sum_{e \in C(S)} x_e \geq 2 & \forall S \subset \{1, \dots, n\}, S \neq \emptyset \\ x_e \in \{0, 1\} & \forall e \in E \end{cases}$$

Supposons que toutes les contraintes d'élimination de sous-tours ne sont pas présentes. Soit \tilde{x} la solution optimale de la relaxation PL.

Le problème de séparation consiste à rechercher une contrainte de sous-tour violée, c'est à dire un ensemble S tel que $\sum_{e \in C(S)} \tilde{x}_e < 2$. En particulier, on peut chercher l'ensemble S^* tel que $\sum_{e \in C(S)} \tilde{x}_e$ est minimum et vérifier si sa valeur dépasse 2. Il s'agit du problème de recherche de la coupe minimale dans un graphe, connu pour être polynomial.

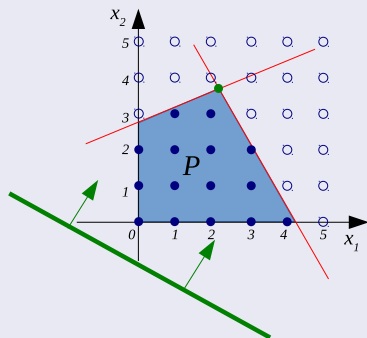
Principe

- Choisir une variable x_i



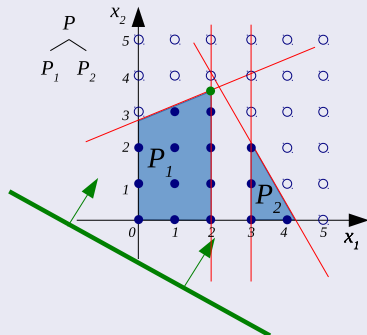
Principe

- Choisir une variable x_i ;
- **Séparer** le problème en deux sous-problèmes selon les valeurs de x_i ;



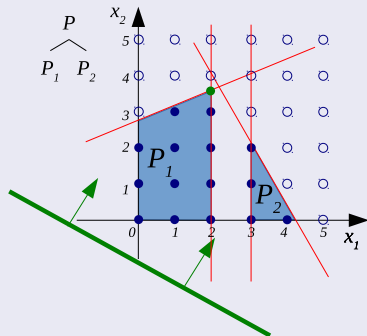
Principe

- Choisir une variable x_i
- **Séparer** le problème en deux sous-problèmes selon les valeurs de x_i
 - en *PLNE*, choisir p tel que les deux sous-problèmes correspondent à $x_i \leq p$ et $x_i \geq p + 1$



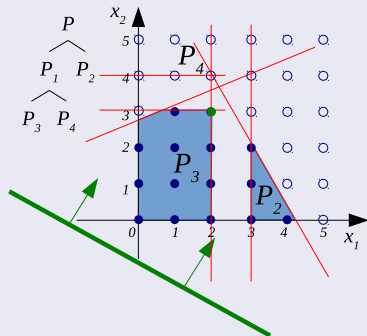
Principe

- Choisir une variable x_i
- **Séparer** le problème en deux sous-problèmes selon la valeur de x_i
 - en *PLNE*, choisir p tel que les deux sous-problèmes correspondent à $x_i \leq p$ et $x_i \geq p + 1$
 - en *PL 0-1*, les deux sous-problèmes correspondent à $x_i = 0$ et $x_i = 1$



Principe

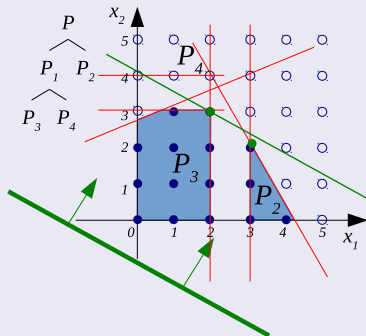
- Choisir une variable x_i
 - **Séparer** le problème en deux sous-problèmes selon les valeurs de x_i
 - en *PLNE*, choisir p tel que les deux sous-problèmes correspondent à $x_i \leq p$ et $x_i \geq p + 1$
 - en *PL 0-1*, les deux sous-problèmes correspondent à $x_i = 0$ et $x_i = 1$
- ⇒ forme arborescente : chaque nœud est un sous-problème



Principe

- Choisir une variable x_i
- **Séparer** le problème en deux sous-problèmes selon la valeur de x_i
 - en *PLNE*, choisir p tel que les deux sous-problèmes correspondent à $x_i \leq p$ et $x_i \geq p + 1$
 - en *PL 0-1*, les deux sous-problèmes correspondent à $x_i = 0$ et $x_i = 1$

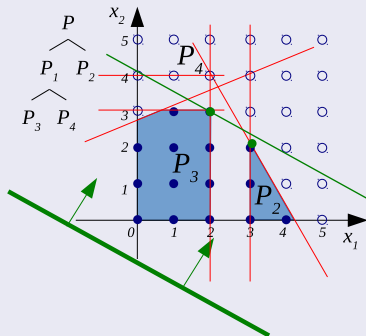
⇒ forme arborescente : chaque nœud est un sous-problème
- Supprimer un sous-problème P_s^E grâce à une **évaluation** par excès de P_s :



Principe

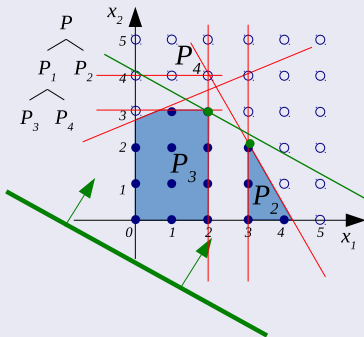
- Choisir une variable x_i
- **Séparer** le problème en deux sous-problèmes selon la valeur de x_i
 - en *PLNE*, choisir p tel que les deux sous-problèmes correspondent à $x_i \leq p$ et $x_i \geq p + 1$
 - en *PL 0-1*, les deux sous-problèmes correspondent à $x_i = 0$ et $x_i = 1$

⇒ forme arborescente : chaque nœud est un sous-problème
- Supprimer un sous-problème P_s^E grâce à une **évaluation** par excès de P_s :
 - 1 Si $\overline{eval}(P_s^E)$ est un majorant du coût des solutions de P_s^E (cas d'une maximisation) ; par exemple solution optimale du problème P_s^E relâché
 - 2 Si on connaît une solution entière x de P^E dont le coût est \tilde{z}



Principe

- Choisir une variable x_i
- **Séparer** le problème en deux sous-problèmes selon les valeur de x_i
 - en *PLNE*, choisir p tel que les deux sous problèmes correspondent à $x_i \leq p$ et $x_i \geq p + 1$
 - en *PL 0-1*, les deux sous-problèmes correspondent à $x_i = 0$ et $x_i = 1$
- ⇒ forme arborescente : chaque nœud est un sous-problème
- Supprimer un sous-problème P_s^E grâce à une **évaluation** par excès de P_s :
 - 1 Si $\overline{eval}(P_s^E)$ est un majorant du coût des solutions de P_s^E (cas d'une maximisation) ; par exemple solution optimale du problème P_s^E relâché
 - 2 Si on connaît une solution entière x de P^E dont le coût est \tilde{z}
- ⇒ Si $eval(P_s^E) \leq \tilde{z}$, alors le sous-problème P_s^E n'est pas intéressant (ne contient pas la solution optimale).



Évaluation d'un nœud : on applique l'algorithme de résolution du PL (et on ajoute éventuellement des coupes). Trois cas d'interruption de la séparation :

- **Optimalité locale** : la solution de la relaxation (+ coupes) est entière. C'est la meilleure solution entière accessible à partir du nœud.
- **Dominance** (voir slide précédent) : la relaxation (+ coupes) donne une borne supérieure inférieure ou égale à la meilleure borne inférieure connue (problème de maximisation). La meilleure solution entière accessible à partir du nœud ne peut améliorer la solution dont on dispose déjà.
- **Infaisabilité** : la relaxation (+ coupes) est irréalisable. Il n'existe aucune solution entière accessible à partir du nœud.

Remarque : en complément de l'évaluation on peut exécuter à chaque nœud des heuristiques tentant d'obtenir des solutions entières à partir de la solution continue du nœud pour améliorer la borne inférieure (et permettre de déclencher la condition de dominance plus tôt).

Si aucune de ces conditions n'est vérifiée on doit séparer le nœud.

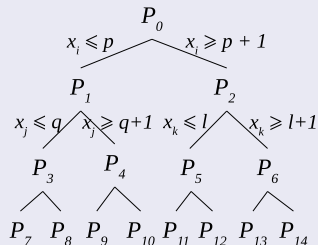
Risque

Le nombre de sous-problèmes générés est exponentiel par rapport à la taille du problème.

- exemple PL en 0-1 à trente variables : $2^{30} \sim 10^9$ branchements
- l'énumération explicite prendrait des siècles

3 degrés de liberté des méthodes par séparation et évaluation

- Quelle fonction d'évaluation par excès choisir ?



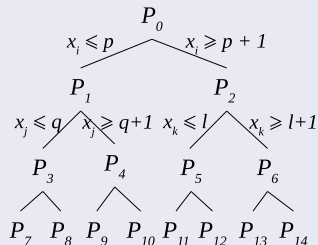
Risque

Le nombre de sous-problèmes générés est exponentiel par rapport à la taille du problème.

- exemple *PL* en 0-1 à trente variables : $2^{30} \sim 10^9$ branchements
- l'énumération explicite prendrait des siècles

3 degrés de liberté des méthodes par séparation et évaluation

- Quelle fonction d'évaluation par excès choisir ?
 - exemple : solution optimale continue du *PLNE*



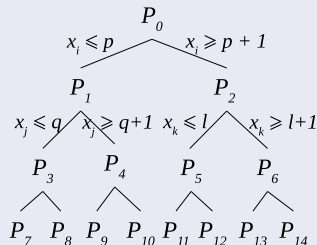
Risque

Le nombre de sous-problèmes générés est exponentiel par rapport à la taille du problème.

- exemple *PL* en 0-1 à trente variables : $2^{30} \sim 10^9$ branchements
- l'énumération explicite prendrait des siècles

3 degrés de liberté des méthodes par séparation et évaluation

- Quelle fonction d'évaluation par excès choisir ?
 - exemple : solution optimale continue du *PLNE*
 - compromis qualité/rapidité



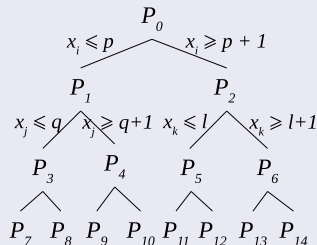
Risque

Le nombre de sous-problèmes générés est exponentiel par rapport à la taille du problème.

- exemple *PL* en 0-1 à trente variables : $2^{30} \sim 10^9$ branchements
- l'énumération explicite prendrait des siècles

3 degrés de liberté des méthodes par séparation et évaluation

- Quelle fonction d'évaluation par excès choisir ?
 - exemple : solution optimale continue du *PLNE*
 - compromis qualité/rapidité
- Quelle variable choisir pour la séparation ?



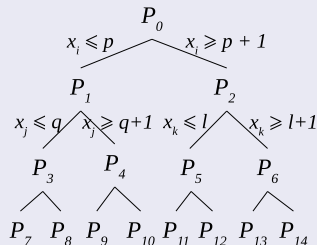
Risque

Le nombre de sous-problèmes générés est exponentiel par rapport à la taille du problème.

- exemple *PL* en 0-1 à trente variables : $2^{30} \sim 10^9$ branchements
- l'énumération explicite prendrait des siècles

3 degrés de liberté des méthodes par séparation et évaluation

- Quelle fonction d'évaluation par excès choisir ?
 - exemple : solution optimale continue du *PLNE*
 - compromis qualité/rapidité
- Quelle variable choisir pour la séparation ?
 - ordre statique ou dynamique



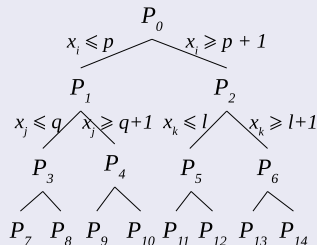
Risque

Le nombre de sous-problèmes générés est exponentiel par rapport à la taille du problème.

- exemple *PL* en 0-1 à trente variables : $2^{30} \sim 10^9$ branchements
- l'énumération explicite prendrait des siècles

3 degrés de liberté des méthodes par séparation et évaluation

- Quelle fonction d'évaluation par excès choisir ?
 - exemple : solution optimale continue du *PLNE*
 - compromis qualité/rapidité
- Quelle variable choisir pour la séparation ?
 - ordre statique ou dynamique
 - autres règle de séparation : $x_1 = x_2$ et $x_1 \neq x_2$



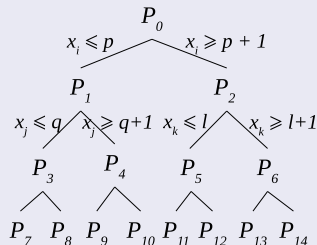
Risque

Le nombre de sous-problèmes générés est exponentiel par rapport à la taille du problème.

- exemple *PL* en 0-1 à trente variables : $2^{30} \sim 10^9$ branchements
- l'énumération explicite prendrait des siècles

3 degrés de liberté des méthodes par séparation et évaluation

- Quelle fonction d'évaluation par excès choisir ?
 - exemple : solution optimale continue du *PLNE*
 - compromis qualité/rapidité
- Quelle variable choisir pour la séparation ?
 - ordre statique ou dynamique
 - autres règle de séparation : $x_1 = x_2$ et $x_1 \neq x_2$
- Quel sous-problème résoudre en premier ?
C'est-à-dire : comment parcourir l'arbre ?



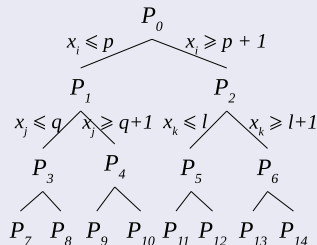
Risque

Le nombre de sous-problèmes générés est exponentiel par rapport à la taille du problème.

- exemple *PL* en 0-1 à trente variables : $2^{30} \sim 10^9$ branchements
- l'énumération explicite prendrait des siècles

3 degrés de liberté des méthodes par séparation et évaluation

- Quelle fonction d'évaluation par excès choisir ?
 - exemple : solution optimale continue du *PLNE*
 - compromis qualité/rapidité
- Quelle variable choisir pour la séparation ?
 - ordre statique ou dynamique
 - autres règle de séparation : $x_1 = x_2$ et $x_1 \neq x_2$
- Quel sous-problème résoudre en premier ?
C'est-à-dire : comment parcourir l'arbre ?
 - **parcours en profondeur d'abord** : on choisit le sous-problème le plus bas dans l'arbre ; ie on descend dans l'arbre afin de trouver une solution réalisable le plus rapidement possible.



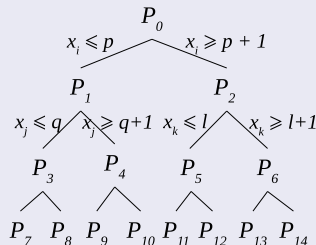
Risque





Le nombre de sous-problèmes générés est exponentiel par rapport à la taille du problème.





- exemple *PL* en 0-1 à trente variables : $2^{30} \sim 10^9$ branchements
- l'énumération explicite prendrait des siècles

3 degrés de liberté des méthodes par séparation et évaluation

- Quelle fonction d'évaluation par excès choisir ?
 - exemple : solution optimale continue du *PLNE*
 - compromis qualité/rapidité
- Quelle variable choisir pour la séparation ?
 - ordre statique ou dynamique
 - autres règle de séparation : $x_1 = x_2$ et $x_1 \neq x_2$
- Quel sous-problème résoudre en premier ?
C'est-à-dire : comment parcourir l'arbre ?
 - **parcours en profondeur d'abord** : on choisit le sous-problème le plus bas dans l'arbre ; ie on descend dans l'arbre afin de trouver une solution réalisable le plus rapidement possible.
 - **Meilleur d'abord** : meilleure évaluation par excès [...]



-  Alain Haït.
Programmation linéaire en nombres entiers
Support de cours ISAE - SUPAERO, 2014.
-  Alexandre Gondran
Optimisation Combinatoire
Support de cours ENAC, 2014.
-  Michel Minoux.
Programmation mathématique.
Lavoisier, 2008, 2^e édition.
-  Dominique de Werra, Thomas Liebling, Jean-François Hêche.
Recherche opérationnelle pour ingénieurs.
Presses polytechniques et universitaires romandes, 2003.
-  Stephen Bradley, Arnaldo Hax, Thomas Magnanti
Applied mathematical programming.
MIT Press, Addison-Wesley, 1977.
<http://web.mit.edu/15.053/www/>

-  Laurence Wosley
Integer programming.
Wiley -Interscience, 1998.
-  François Soumis
Modélisation en recherche opérationnelle.
Support de cours École Polytechnique de Montréal.
-  Michel Gamache
Recherche opérationnelle minière.
Support de cours École Polytechnique de Montréal.
-  Andrea Lodi
Problem-solving by Mixed-Integer Programming.
Présentation à ROADEF 2014.



R.E. Burkard, *Time-slot assignment for TDMA-systems*, Computing **35** (1985), no. 2, 99–112, <http://link.springer.com/article/10.1007%2FBF02260498>.



L. Barbulescu, J.-P. Watson, L.D. Whitley, and A.E. Howe, *Scheduling space-ground communications for the air force satellite control network*, Journal of Scheduling **7** (2004), no. 1, 7–34, <http://www.cs.colostate.edu/sched/pubs/jos03.pdf>.



Max Cerf, *Multiple space debris collecting mission - debris selection and trajectory optimization*, Tech. Report CoRR abs/1107.0192, EADS Astrium Space Transportation, 2011, <http://arxiv.org/abs/1107.0192>.



E. Dupuis, I. Rekleitis, J.-L. Bedwani, T. Lamarche, P. Allard, and W.-H. Zhu, *Over-the-horizon autonomous rover navigation : Experimental results*, International Symposium on Artificial Intelligence, Robotics and Automation in Space – i-SAIRAS 2008, 2008, <http://robotics.estec.esa.int/i-SAIRAS/isairas2008/Proceedings/SESSION%209/m042-Dupuis.pdf>.



A. Dutta, *Optimal cooperative and non-cooperative peer-to-peer maneuvers for refueling satellites in circular constellations*, Ph.D. thesis, School of Aerospace Engineering, Georgia Institute of Technology, 2009, https://smartech.gatech.edu/bitstream/handle/1853/28082/Dutta_Atri_200905_phd.pdf.



A. Haït, T. Siméon, and M. Taix, *Algorithms for rough terrain trajectory planning*, *Advanced Robotics* **16** (2002), no. 8, 673–699,
homepages.laas.fr/nic/Papers/02hait.ps.gz.



D.B. Manner, *TROUBLE 3 : A fault diagnostic expert system for space station freedom's power system*, Tech. Report NASA Technical report 19910014913, NASA, 1990,
<http://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/19910014913.pdf>.



M. Vasquez and J.-K. Hao, *A logic-constrained knapsack formulation and a tabu algorithm for the daily photograph scheduling of an earth-observing satellite*, *Computational Optimization and Applications* **20** (2001), 137–157,
<http://www.info.univ-angers.fr/pub/hao/papers/JCOAP.pdf>.