Cours n°7: Algorithmes génériques et implémentations de résolution en Programmation Linéaire en (Nombres Entiers)

Nicolas DUPIN

`https://github.com/ndupin/ORteaching`
`http://nicolasdupin2000.wixsite.com/research`

version du March 19, 2022

Issu et étendu d'enseignements donnés à l'ENSTA Paris, Polytech' Paris-Saclay, et à l'université Paris-Saclay

Rappels: modélisation PLNE

- De nombreux problèmes industriels et académiques d'optimisation se modélisent en PL/PLNE.
- 1 contrainte = 1 exigence, les exigences s'ajoutent indépendamment, permet des itérations agiles pour modéliser un problème avec une description du problème qui s'affine au cours des itérations.
- Définition mathématique PL/PLNE décrit bien un problème (description littérale peut contenir des ambiguités).
- La linéarité n'est pas une hypothèse si restrictive en PLNE, linéarisations en rajoutant des variables discrètes possibles pour des contraintes quadratiques, logiques ...

$\Longrightarrow$ Comment résoudre de manière générique un PL ou PLNE?

$\Longrightarrow$ Quels outils (logiciels/bibliothèques) peut on utiliser?

Extension de la résolution du sac à dos

Après le cours sur le sac à dos, on va voir comment les algorithmes et opérateurs définis pour le sac à dos s'étendent à tout PLNE:

- Résolution de relaxation continue: algo glouton généralisé par des algorithmes génériques de résolution PL (comme les algos de simplexes)
- Heuristiques primales comme glouton discret et Kernel Search? Ne se généralise pas bien, trouver une solution réalisable à un PLNE général définit un pb NP-complet
- Algorithme de Branch&Bound : similaire avec extension des opérateurs de base (résolution PL et branchements), de nombreux paramètres génériques.
- Coupes : généralisation des coupes de couvertures
- Démarrage à chaud de relaxation continue: avec le simplexe dual!
- Heuristiques primales à l'intérieur de l'algo B&B: un autre type de démarrage à chaud avec un impact significatif

Plan

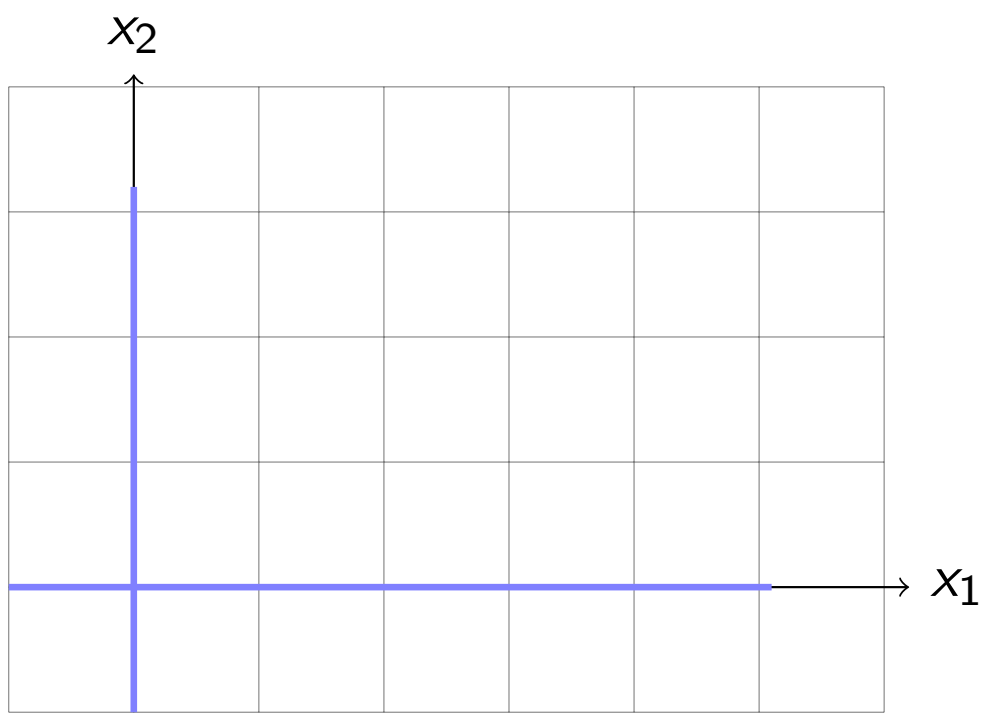
Plan

Rappel: résolution graphique d'un PL à deux variables

Soit le PL suivant:

$$\begin{cases} Z^* = max 1200x_1 + 1500x_2 \\ \quad 4x_1 + 2x_2 \leqslant 12 \\ \quad 2x_1 + 5x_2 \leqslant 10 \\ \quad x_1, x_2 \geq 0 \end{cases}$$

On trace en bleu les contraintes.

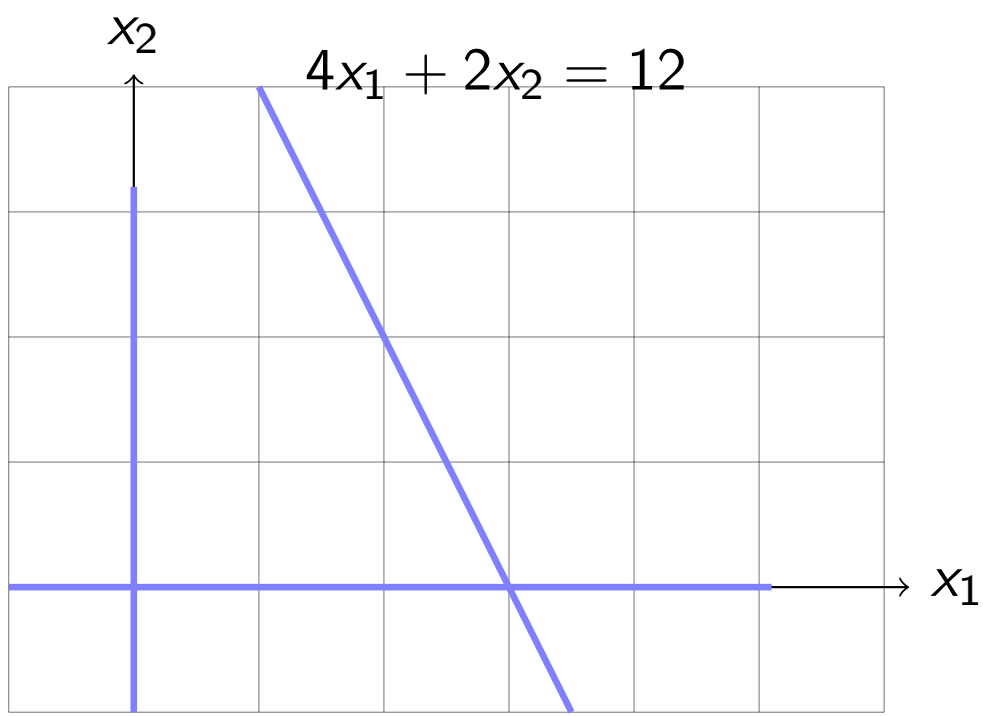


Rappel: résolution graphique d'un PL à deux variables

Soit le PL suivant:

$$\begin{cases} Z^* = max 1200x_1 + 1500x_2 \\ \quad 4x_1 + 2x_2 \leqslant 12 \\ \quad 2x_1 + 5x_2 \leqslant 10 \\ \quad x_1, x_2 \geq 0 \end{cases}$$

On trace en bleu les contraintes.

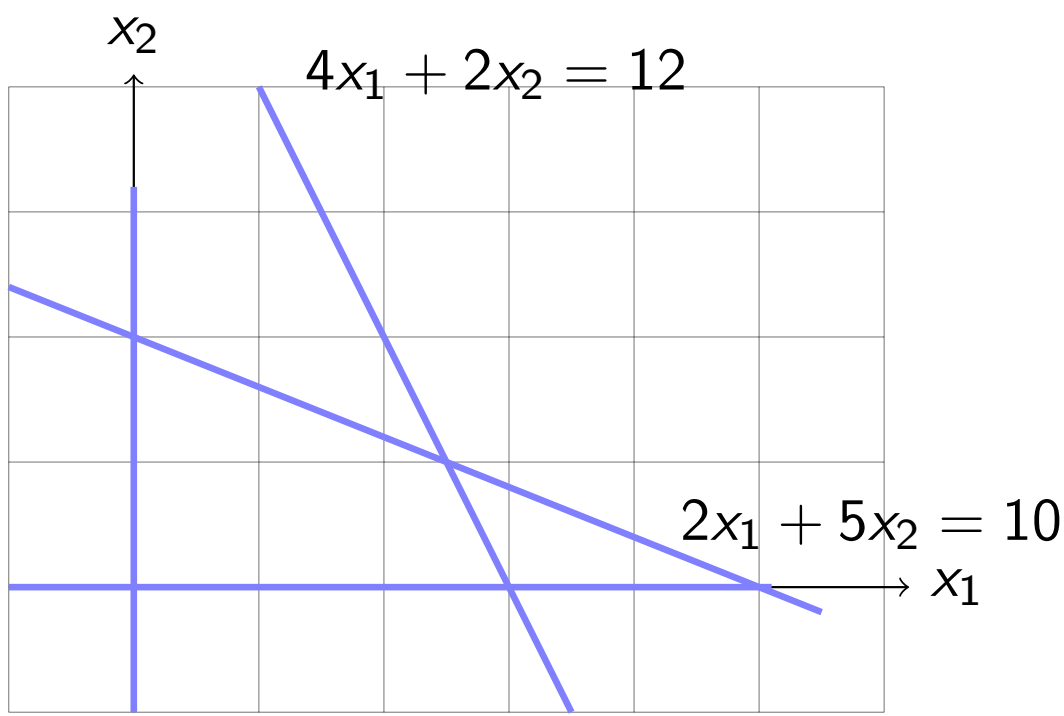


Rappel: résolution graphique d'un PL à deux variables

Soit le PL suivant:

$$\begin{cases} Z^* = max 1200x_1 + 1500x_2 \\ \quad 4x_1 + 2x_2 \leqslant 12 \\ \quad 2x_1 + 5x_2 \leqslant 10 \\ \quad x_1, x_2 \geq 0 \end{cases}$$

On trace en bleu les contraintes.



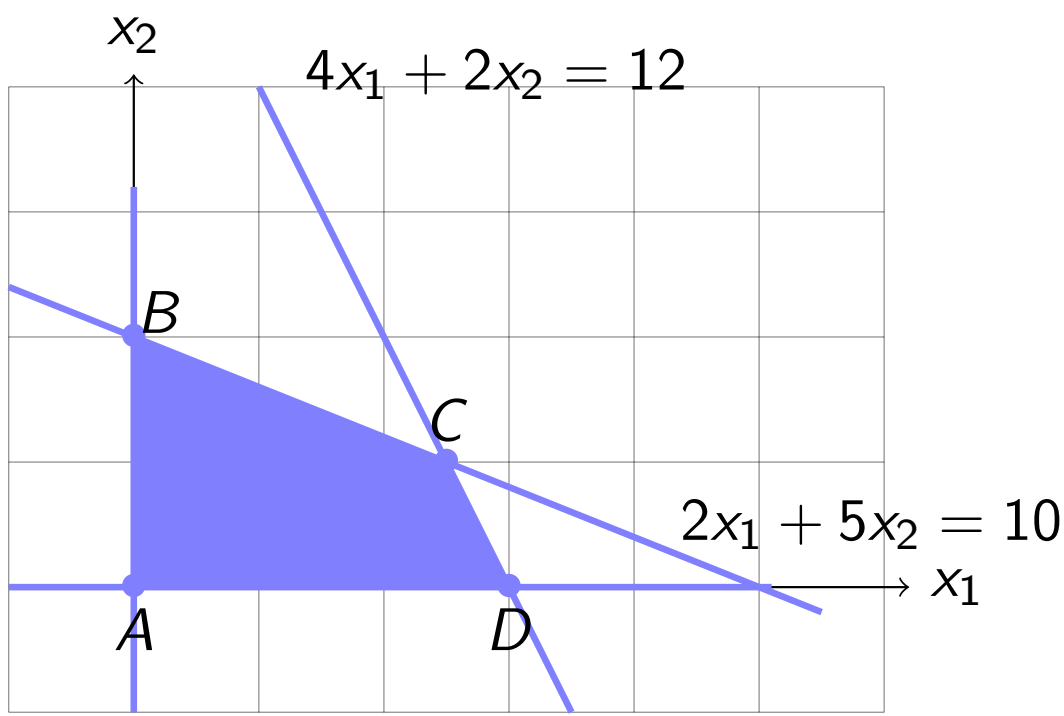
Rappel: résolution graphique d'un PL à deux variables

Soit le PL suivant:

$$\begin{cases} Z^* = max 1200x_1 + 1500x_2 \\ 4x_1 + 2x_2 \leqslant 12 \\ 2x_1 + 5x_2 \leqslant 10 \\ x_1, x_2 \geq 0 \end{cases}$$

Domaine des solutions réalisables: ABCD

On trace en bleu les contraintes.

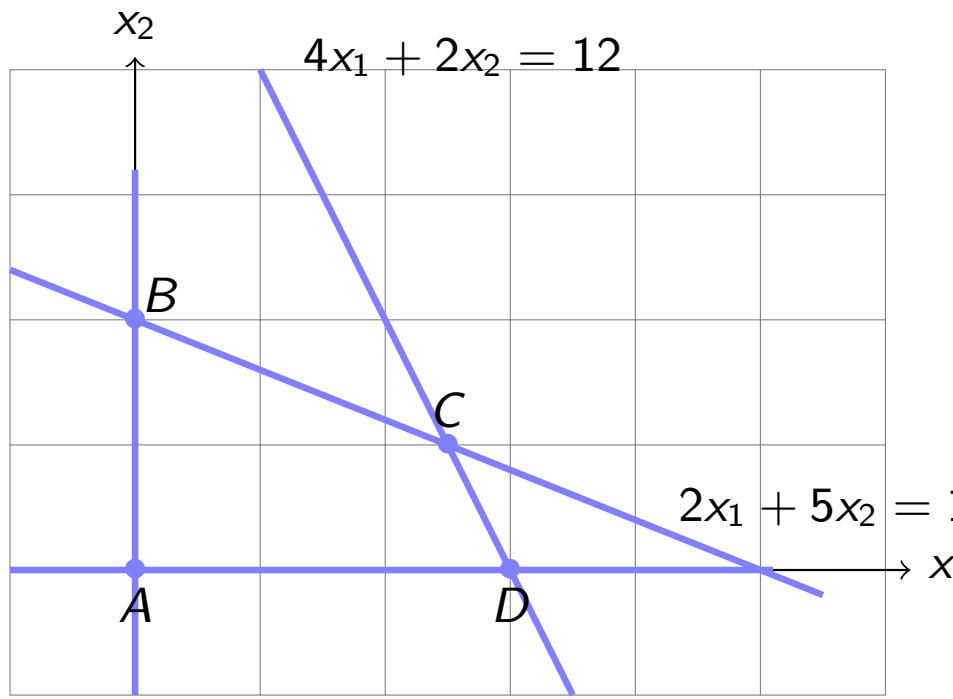


Rappel: résolution graphique d'un PL à deux variables

$$\begin{cases} Z^* = max 1200x_1 + 1500x_2 \\ 4x_1 + 2x_2 \leqslant 12 \\ 2x_1 + 5x_2 \leqslant 10 \\ x_1, x_2 \geq 0 \end{cases}$$

La fonction objectif définit des lignes de niveaux, de points ayant le même coût par la fonction objectif

Les lignes de niveaux sont parallèles

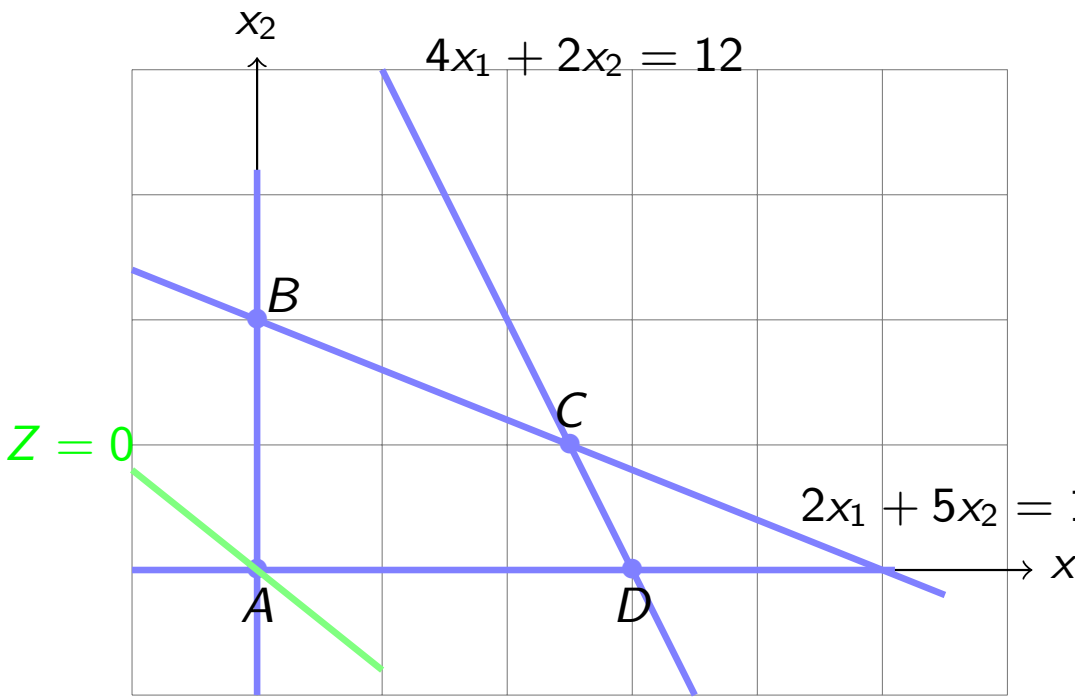


Rappel: résolution graphique d'un PL à deux variables

$$\begin{cases} Z^* = max 1200x_1 + 1500x_2 \\ \quad 4x_1 + 2x_2 \leqslant 12 \\ \quad 2x_1 + 5x_2 \leqslant 10 \\ \quad x_1, x_2 \geq 0 \end{cases}$$

La fonction objectif définit des lignes de niveaux, de points ayant le même coût par la fonction objectif

Les lignes de niveaux sont parallèles

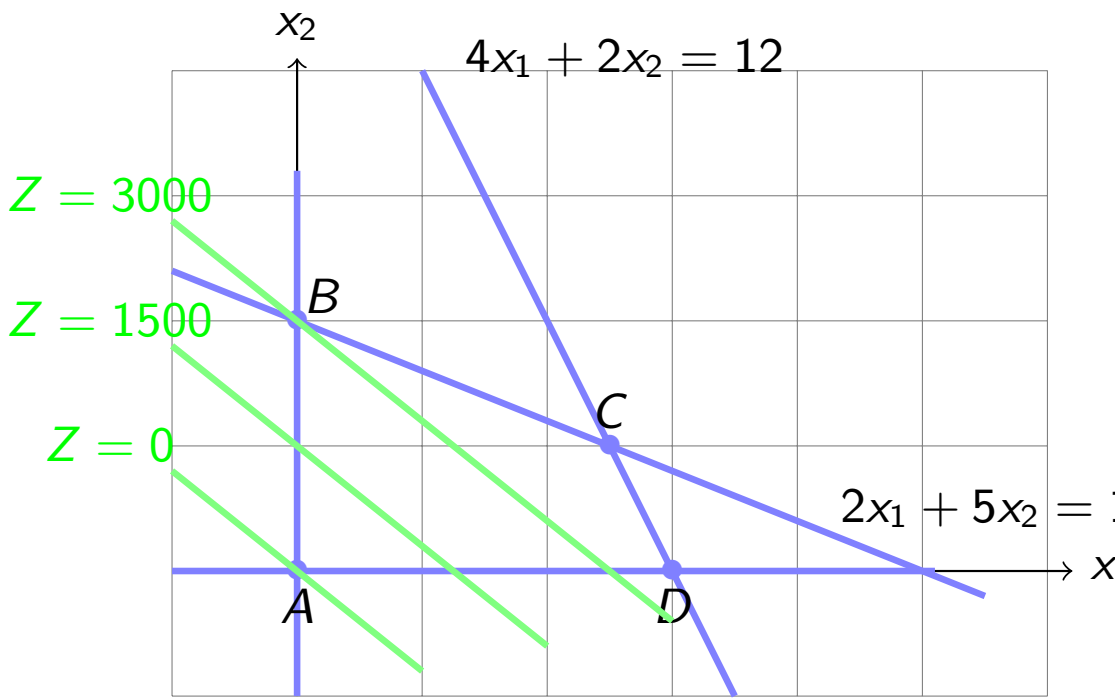


Rappel: résolution graphique d'un PL à deux variables

$$\begin{cases} Z^* = max 1200x_1 + 1500x_2 \\ 4x_1 + 2x_2 \leqslant 12 \\ 2x_1 + 5x_2 \leqslant 10 \\ x_1, x_2 \geq 0 \end{cases}$$

La fonction objectif définit des lignes de niveaux, de points ayant le même coût par la fonction objectif

Les lignes de niveaux sont parallèles

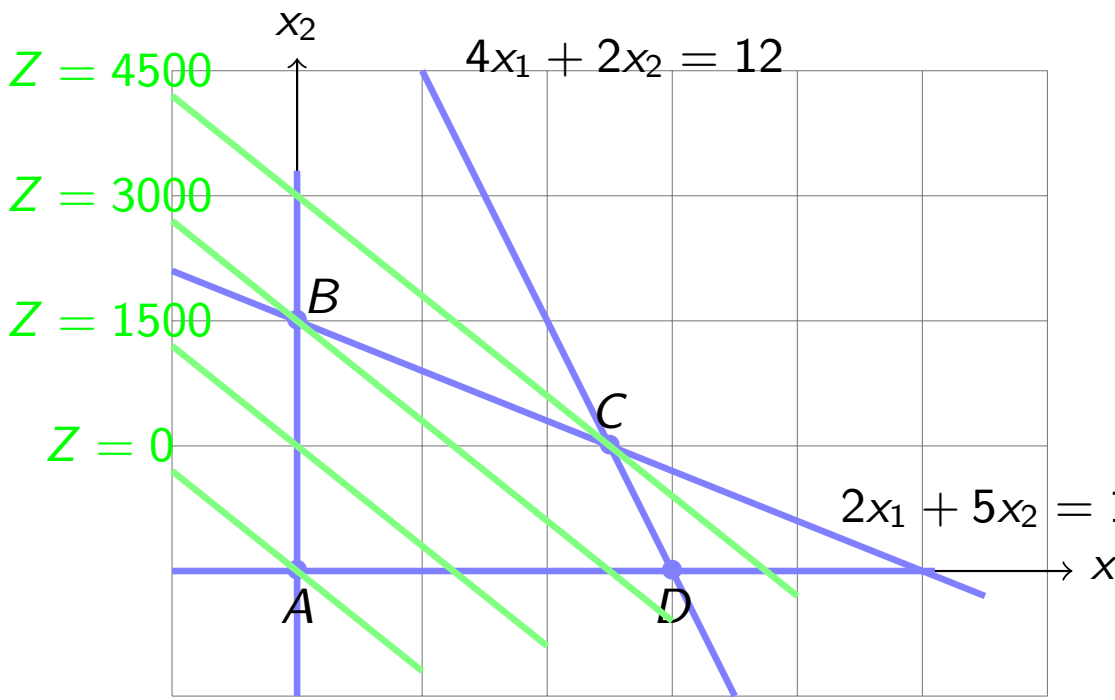


Rappel: résolution graphique d'un PL à deux variables

$$\begin{cases} Z^* = max 1200x_1 + 1500x_2 \\ 4x_1 + 2x_2 \leqslant 12 \\ 2x_1 + 5x_2 \leqslant 10 \\ x_1, x_2 \geq 0 \end{cases}$$

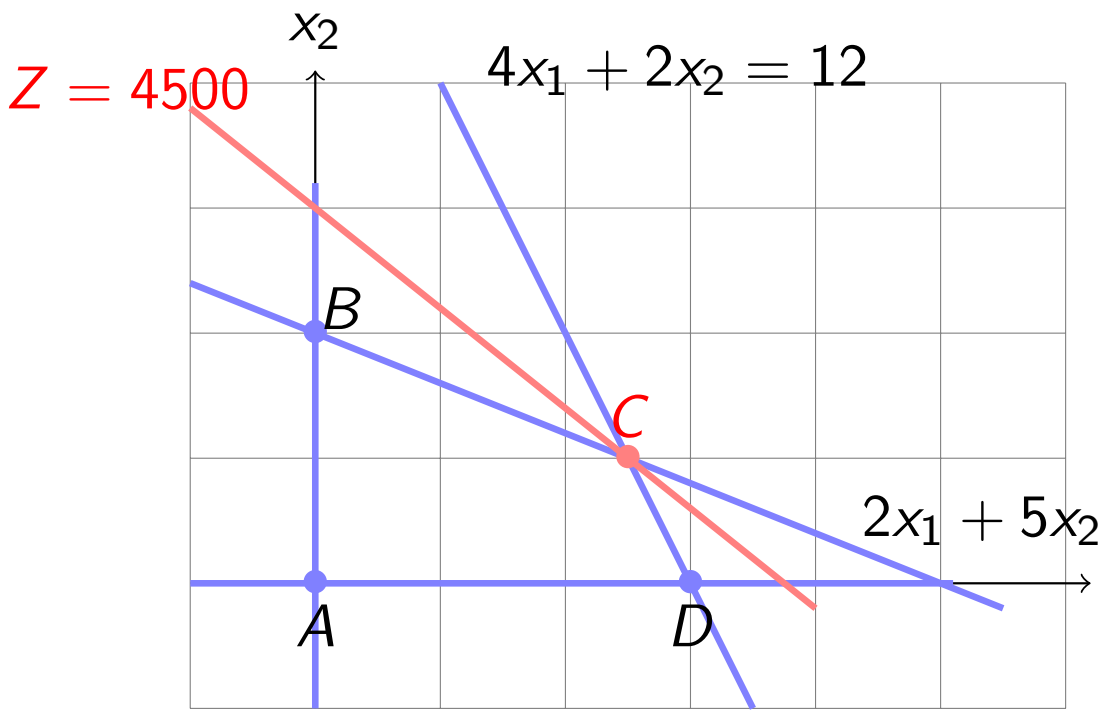
La fonction objectif définit des lignes de niveaux, de points ayant le même coût par la fonction objectif

Les lignes de niveaux sont parallèles



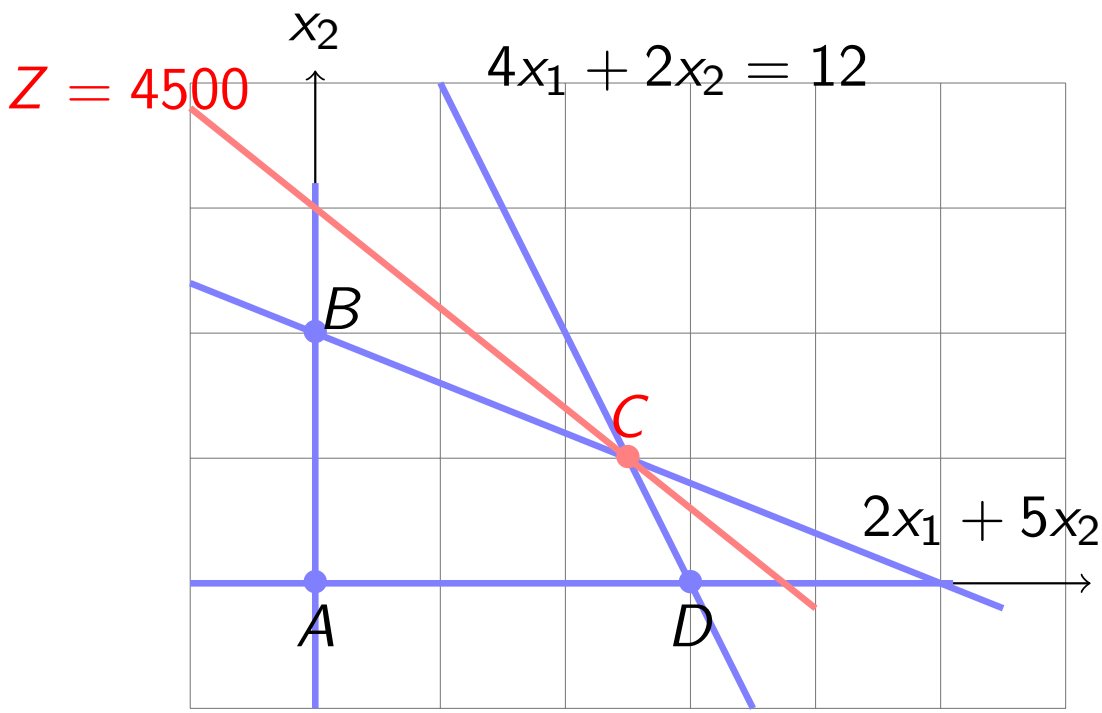
Rappel: résolution graphique d'un PL à deux variables

$$\begin{cases} Z^* = max 1200x_1 + 1500x_2 \\ \quad 4x_1 + 2x_2 \leqslant 12 \\ \quad 2x_1 + 5x_2 \leqslant 10 \\ \quad x_1, x_2 \geq 0 \end{cases}$$



Rappel: résolution graphique d'un PL à deux variables

$$\begin{cases} Z^* = max 1200x_1 + 1500x_2 \\ \quad 4x_1 + 2x_2 \leqslant 12 \\ \quad 2x_1 + 5x_2 \leqslant 10 \\ \quad x_1, x_2 \geq 0 \end{cases}$$

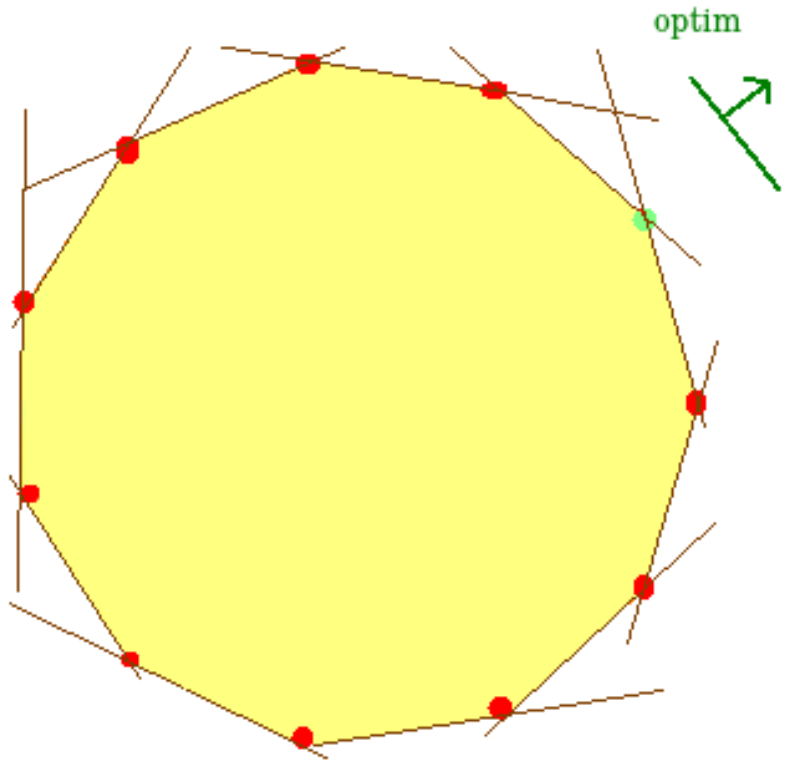


Solution optimale:

$Z^* = 4500$

$x_1^* = 2.5,\ x_2^* = 1$

Illustration d'un PL en 2D, point de vue géométrique



$\Longrightarrow$ avec des contraintes d'inégalités, l'ensemble des solutions réalisables définit un polygone convexe.

$\Longrightarrow$ Suivant la direction de la fonction objectif, l'ensemble des solutions optimales est soit un point extrême, un "sommet", soit une arête tout entière.

Dans le cas des PL, le cas dégénéré avec des solutions optimales le long d'une arête survient quand la fonction objectif définit des ligne de niveaux parallèles à la contrainte active définissant l'arête.

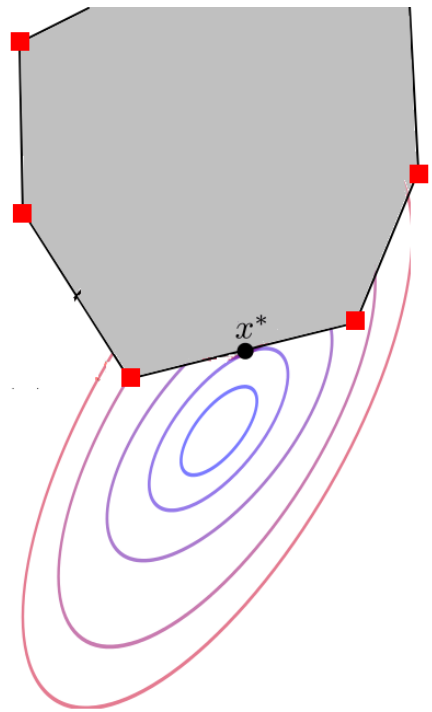
Est-ce général en optimisation sous contrainte en 2D?

$\Longrightarrow$ Un point extrême, un "sommet", est toujours solution d'un PL à deux variables.

$\Longrightarrow$ Finalement, même si l'ensemble des solutions réalisables est un ensemble infini (et indénombrable), un PL se résoudrait en énumérant un nombre fini de possibilité, les sommets!

Cela est vrai car objectif et contraintes linéaires.

N.B: Avec des contraintes linéaires et un objectif non linéaire, on peut avoir une unique solution optimale sur une arrête mais pas sur un sommet du polyèdre des contraintes



De la dimension 2 à la dimension n

La fonction objectif est un hyper-plan et nos contraintes sont aussi des hyperplans ou bien des demi-espaces.

Définition (Polyèdre)

Un polyèdre est une intersection d'un nombre fini de demi espaces affines fermés. En d'autres termes $P \subset \mathbb{R}^n$ est un polyèdre s'il existe un entier $p \in \mathbb{N}$ et une matrice de taille $p \times n$, $b \in \mathbb{R}^p$ tels quel $P = \{x \in \mathbb{R}^n, A.x \leqslant b\}$.

Un polyèdre est un sous-ensemble convexe et fermé de $\mathbb{R}^n$.

Définition (Polytopes)

Un polytope est un polyèdre borné.

$\Longrightarrow$ Intuition géométrique: l'ensemble des solutions réalisables d'un PL est un polyèdre, avec des variables bornées (comme dans bcp d'applications), on a un polytope.

Enveloppe convexe

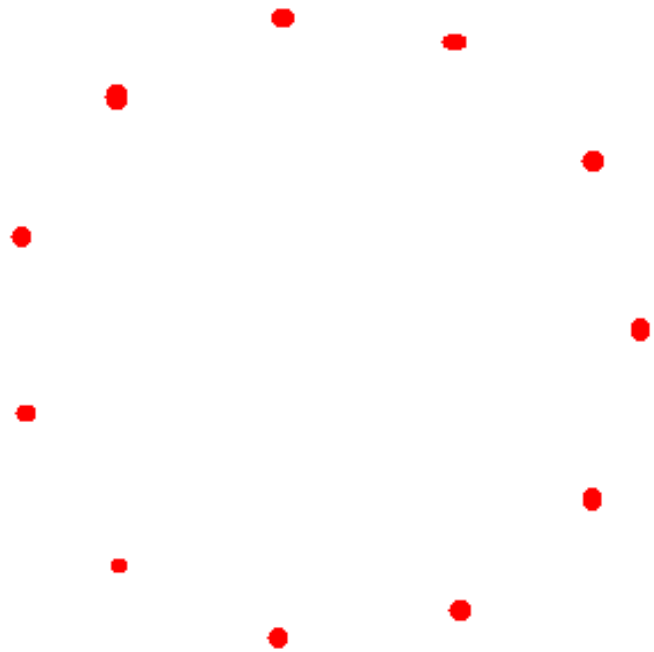
Définition (Enveloppe convexe)

Soit A une partie de $\mathbb{R}^n$. L'enveloppe convexe de A est le plus petit convexe de $\mathbb{R}^n$ contenant A (au sens de l'inclusion):

$$conv(A) = \bigcap_{A \subset B \subset \mathbb{R}^n : B \text{ convexe}} B$$

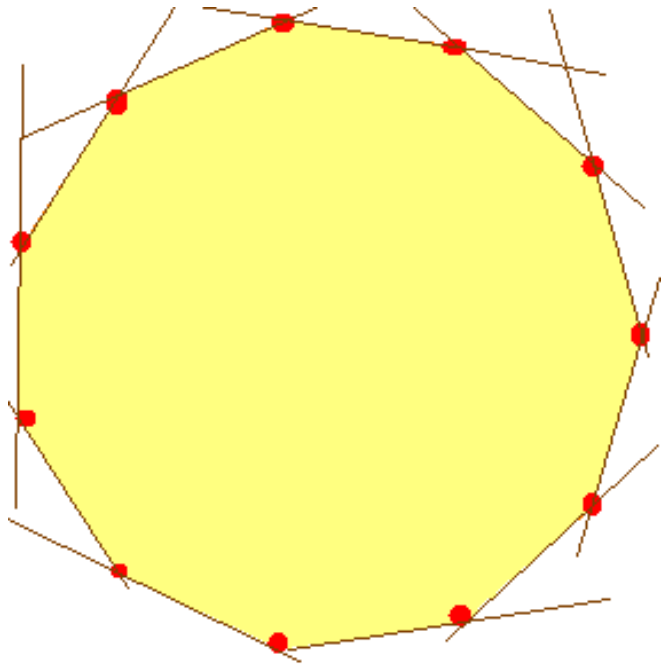
$conv(A)$ est l'ensemble des combinaisons convexes de familles finies d'éléments de A: $conv(A) = \{x \in \mathbb{R}^n | x = \sum_{i=1}^{m} \lambda_i a_i\}$ pour un certain $m \in \mathbb{N}$, et des familles $(a_1, \ldots, a_m) \in A^m$ et $(\lambda_1, \ldots, \lambda_m) \in \mathbb{R}_+^m$, telles que $\sum_i \lambda_i = 1$,
L'enveloppe convexe d'un nombre fini de points est un polytope.

Caractérisation d' enveloppe convexe



Enveloppe convexe d'un nombre fini de points a_i? C'est un polyèdre le plus petit convexe contenant ces points, c'est ...

Illustration d' enveloppe convexe



$$conv(A) = \left\{ x \in \mathbb{R}^n | \exists (\lambda_i) \geqslant 0, \sum_{i=1}^{m} \lambda_i = 1,\ x = \sum_{i=1}^{m} \lambda_i a_i \right\}$$

Les points a_i sont caractérisés par $(\lambda_i) \in \mathbb{N}$

Cônes

Définition (Cônes)

Un cône $C \subset \mathbb{R}^n$ est un ensemble stable par multiplication par un scalaire strictement positif, ie pour tout $\alpha > 0$, $C = \alpha.C$
Si C est également un polyèdre, c'est un cône polyédral

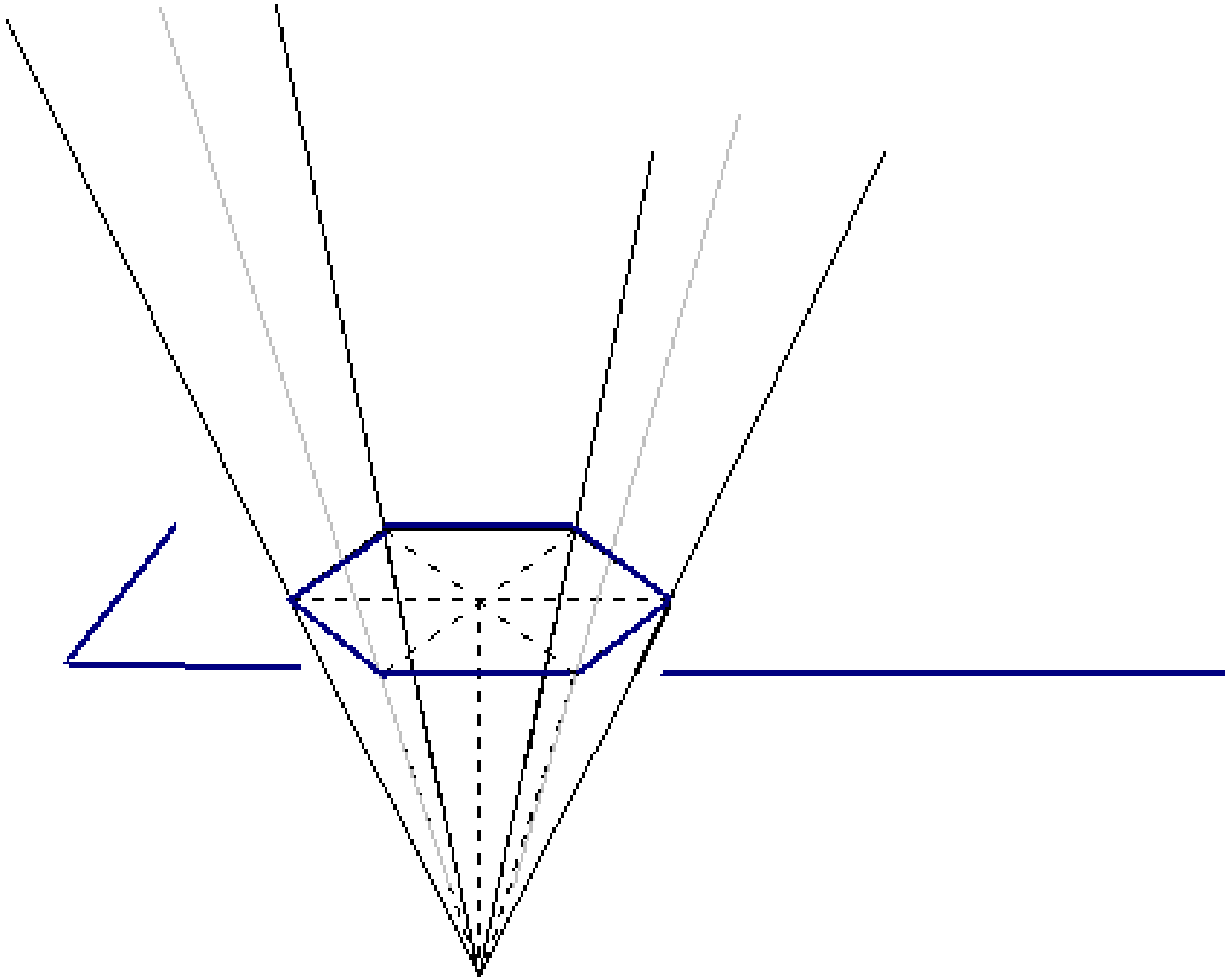
Théorème (Caractérisation des cônes polyédraux)

$C \subset \mathbb{R}^n$ est un cône polyédral si et seulement si C est de la forme $C = \{x \in \mathbb{R}^n, A.x \leqslant 0\}$ où A un matrice de taille $p \times n$, $p \in \mathbb{N}$.

Définition (Cône engendré)

Soit A une partie de $\mathbb{R}^n$. Le cône convexe engendré par A, noté cone(A), est l'ensemble des combinaisons linéaires positives d'un nombre fini de vecteurs de A.

Illustration d'un cône polyédral



Théorème de Minkowski-Weyl

Théorème (Théorème de Minkowski-Weyl)

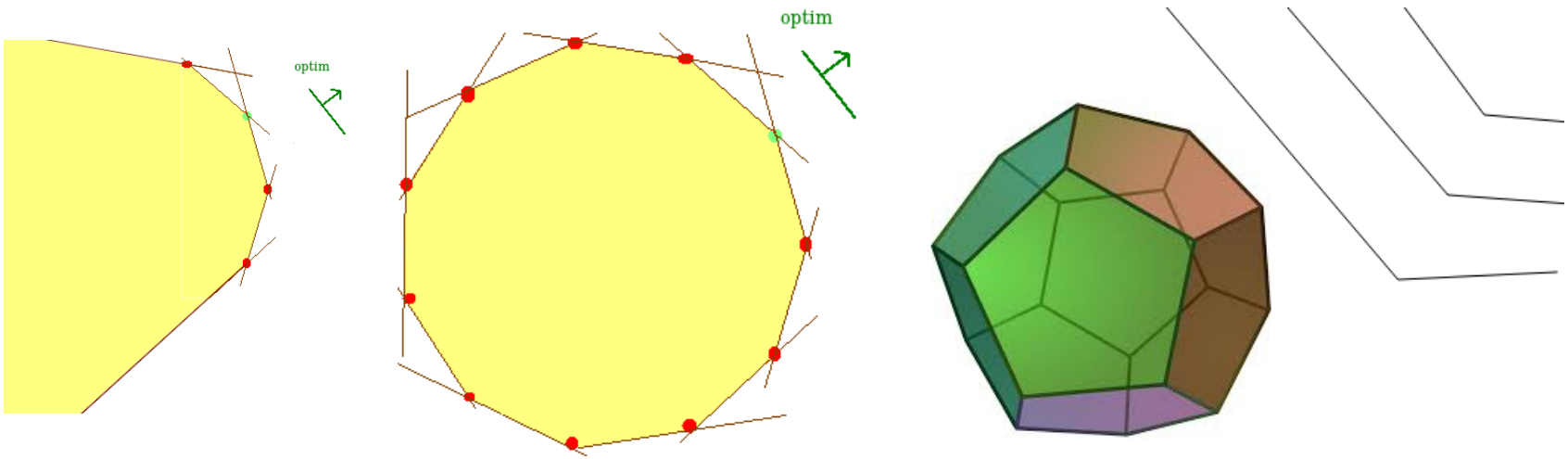
La classe des polyèdres coïncide avec la classe des ensembles de la forme

$$P := conv\{x^i, i \in I\} + cone\{y^j, j \in J\} \tag{1}$$

où $(x^i)_{i\in I}$ et $(y^j)_{j\in J}$ sont des familles finies de vecteurs.
En particulier, la classe des polytopes coïncide avec celle des combinaisons convexes d'un nombre fini de vecteurs.

Illustration géométrique

Pour la dimension n, les solutions réalisables sont plus définies par un polyèdre, les directions d'optimisation et lignes de niveaux sont définies par des hyperplans:



$\Longrightarrow$ On trouve toujours une solution optimale sur un point extrême.

cela ne se représente pas en dimension n, mais l'intuition des dimensions 2 et 3 sont assez représentatives

Points extrêmes d'un polyèdre, caractérisation analytique

définition géométrique d'un point extrême:

Définition (Points extrême d'un ensemble convexe)

Soit C un ensemble convexe de $\mathbb{R}^n$. On dit que $x \in C$ est un point extrême du convexe C si:

$$\left[\exists y, z \in C, \alpha \in]0,1[, x = \alpha y + (1-\alpha)z\right] \Longrightarrow x = y = z$$

Proposition (Caractérisation des points extrême d'un polyèdre)

Soit $P = \{x \in \mathbb{R}^n, A.x \leqslant b\}$ un polyèdre de $\mathbb{R}^n$. Tout point extrême de P est solution d'un système $A'.x = b'$ où A' est une sous-matrice inversible formée d'un sous ensemble de ligne de A, et b' définissant le second membre associé aux lignes sélectionnées de A.

N.B: cette proposition qui se démontre assez directement permet de prouver le théorème de Minkowski-Weyl dans le cadre d'un polytope (cas qui sera suffisant dans toute la suite du cours)

Résolution d'un PL, une vue d'ensemble de l'algorithme de simplexe

Si on a un PL dont l'existence de solution existe (par exemple, cas borné):

- PL= optimisation continue sous contraintes linéaires: on est dans le cadre pour utiliser des algorithmes de gradients projetés.
- Propriétés fondamentale: on trouve une solution optimale d'un PL parmi les sommets du polyèdre définissant les contraintes.
- Application: on peut énumérer tous les sommets pour trouver la valeur optimale? On se ramène à un ensemble fini, alors que l'ensemble de départ initial était infini et indénombrable.
- Algorithme du simplexe: peut être vu comme un cas particulier d' algorithme de gradient projeté à pas optimal spécialisé: le pas optimal mène à un point extrême.
- Intuition géométrique du simplexe: A partir d'un sommet, on prend le meilleur voisin selon la direction d'optimisation.

Rappels: Relaxation continue, avec le simplexe dual

Majoration de relaxation continue, avec la dualité forte:

$$\begin{array}{lllllll} OPT = & \max c.x & \leqslant & \max c.x & = & \min b^T.y \\ s.c: & A.x \leqslant b & s.c: & A.x \leqslant b & s.c: & A^T.y \geqslant c^T \\ & x \in \mathbb{N}^m \times \mathbb{R}_+^p & & x \geqslant 0 & & y \geqslant 0 \end{array} \tag{2}$$

Quand on utilise le simplexe dual, les itérations donnent des évaluations $b^T.y$ de solutions continues, qui convergent vers la borne de relaxation continue, mais de manière décroissante.

Toute évaluation $b^T.y$ avec $y \geqslant 0$ vérifiant $A^T.y \geqslant c^T$ est une borne supérieure du problème discret initial, on en calcule à chaque itération de l'algorithme de simplexe dual!

Si on doit arrêter l'algorithme de simplexe dual avec un critère d'arrêt de temps, on a une borne supérieure utilisable.

Rappels: Bornes primales versus bornes duales

Si on écrit le problème comme une minimisation, on a les résultats équivalents:

• La relaxation continue est une borne inférieure de l'optimum entier.

• L'algorithme de simplexe primal converge vers la relaxation continue, tout en décroissant, les bornes données à toute itération intermédiaire ne sont pas en général des bornes inférieures de l'optimum entier

• L'algorithme de simplexe dual converge vers la relaxation continue, tout en croissant, les bornes données à toute itération intermédiaire sont des bornes inférieures de l'optimum entier.

Les notions de primal/dual s'utilisent de manière générique pour des problèmes de minimisation ou de maximisation:

• borne primale: borne supérieure (resp inférieure) pour un problème de minimisation (resp maximisation), donnée par une heuristique qui trouve des solutions entières réalisables au problème initial.

• borne duale: borne inférieure (resp supérieure) pour un problème de minimisation (resp maximisation), donné par relaxation continue. L'algo de simplexe dual: une solution "dual réalisable" donne une borne duale

Des références

Pour avoir l'algorithme du simplexe écrit de manière générale, les fondements mathématiques et les démonstrations, vous pouvez vous reporter au cours suivant disponible en ligne:

G. ALLAIRE, A. ERN, *Optimisation et contrôle*, cours de l'Ecole Polytechnique.
`http://www.cmap.polytechnique.fr/~allaire/map435/poly435.pdf`

• chapitre 2: fondement généraux de l'optimisation sous contrainte (continue), cas général de la dualité. (vous verrez ça en M1 maths appliquées)

• chapitre 3: algorithmes numériques (de type gradient) (vu en M1 maths appliquées, un cas particulier en IA/ML, back-propagation réseau de neurones)

• chapitre 4: l'optimisation linéaire, la dualité et l'algorithme du simplexe, ça s'appréhende naturellement après les chapitres 2 et 3.

Autre cours de M1 maths appliquées que je vous conseille, détaillé pour la partie algorithme numériques de type gradients, ça peut vous servir pour l'IA:

Aude Rondepierre, *Méthodes numériques pour l'optimisation non linéaire déterministe*, cours de l'INSA Toulouse.
`http://www.math.univ-toulouse.fr/~rondep/CoursTD/polyGMM4.pdf`

Références pour la pratique de l'algorithme du simplexe

Algorithme du simplexe: peut être vu comme un algorithme de gradient projeté spécialisé avec la propriété qu'un point extrême est solution. A partir d'un sommet, on prend le meilleur voisin selon la direction d'optimisation.

$\Longrightarrow$ vision géométrique, analytiquement, ça ressemble à des pivots de Gauss

La vision géométrique/pivot de Gauss a été présentée au CM3. Une version "tableau" du simplexe est couramment enseignée.

Elements de cours sur l'algorithme du simplexe:

`https://www2.mat.ulaval.ca/fileadmin/Cours/MAT-2920/Chapitre3.pdf`

Exemples déroulés de l'algorithme du simplexe:

`http://webia.lip6.fr/~gonzales/teaching/optimisation/cours/cours02.pdf`

`https://www.youtube.com/watch?v=B6xSwsgS4zg`

`https://www.youtube.com/watch?v=I3hXfnhJ_kI`

Algorithmes de résolution d'un PL

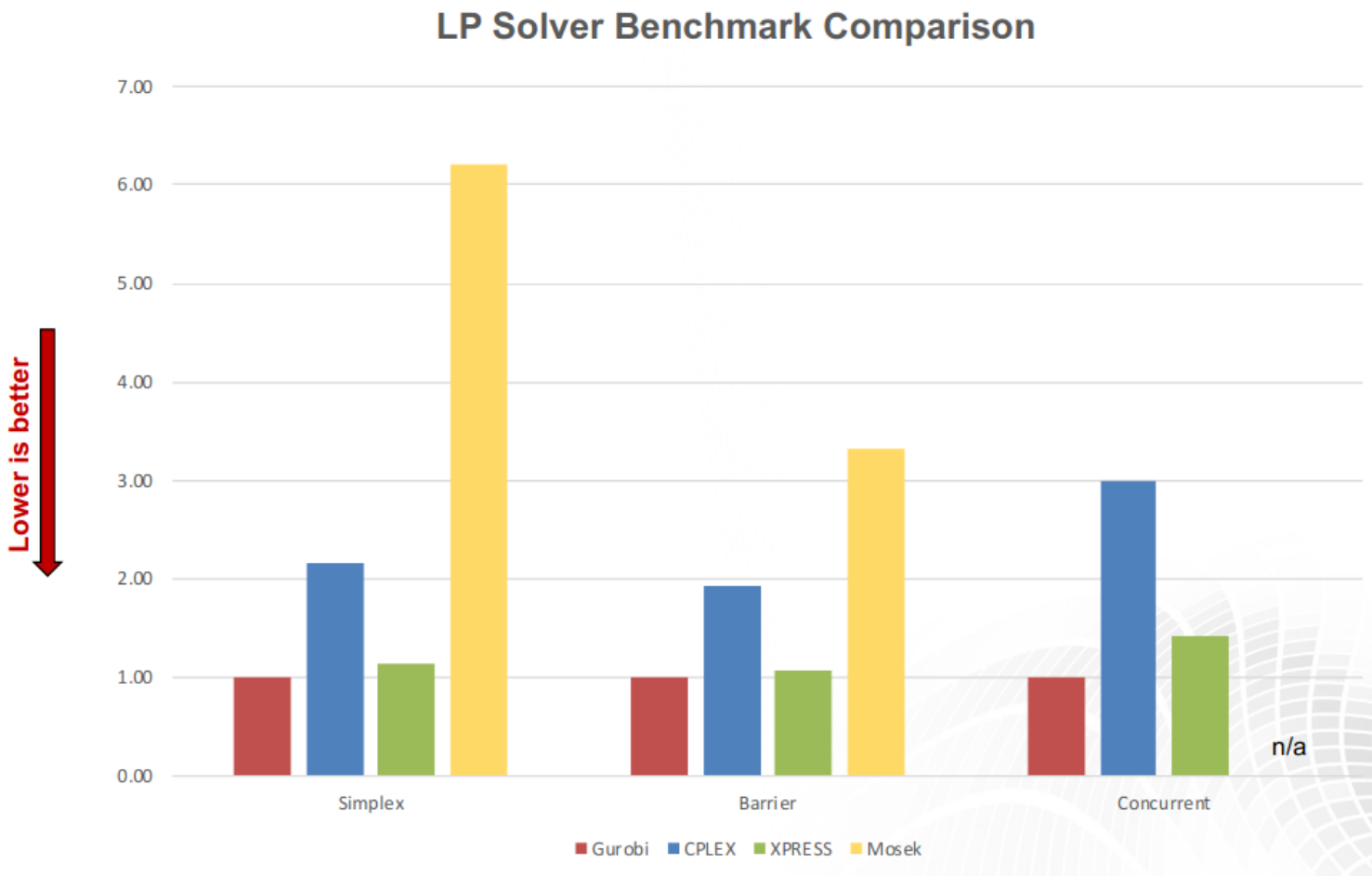
Les solutions optimales d'un PL se trouvent parmi les sommets du polyèdre définissant les contraintes.

- Algos de simplexes: efficace en pratique, il existe des pire cas avec une complexité exponentielle, complexité polynomiale en moyenne (en probabilité). De nombreuses variantes et évolutions du Simplexe.
- Premier algorithme polynomial de résolution PL (et bien plus): méthode de l'ellipsoïde

  Grötschel, M., Lovász, L., & Schrijver, A. (1981). The ellipsoid method and its consequences in combinatorial optimization. Combinatorica, 1(2), 169-197.
- Méthode de l'ellipsoïde: complexité polynomiale, mais très lent en pratique
- Méthode de points intérieurs, algorithme barrières: complexité polynomiale, mais était plus lent en pratique que les simplexes jusqu'à récemment (2010s).

$\Longrightarrow$ Algorithmes codés et à utiliser directement dans les solveurs.

Benchmark de résolution PL, entre solveurs commerciaux



https://www.gurobi.com/pdfs/benchmarks.pdf

Benchmark de résolution PL, entre solveurs libres et Cplex (commercial)

Problem Set	CPLEX	CLP	GLPK	lp_solve[3]	MINOS[4]
Small CCO	0.0	0.1	1.3	19.0	3.1
Infeasible	0.2[1]	3.6	0.7	43.8	16.3
Netlib	9.1	29.5	52.5	14,975.1	3,198.7
Kennington	12.9	16.1	624.3	19,417.5	10,123.8
Large CCO	13.0	19.0	108.4	3,175.8	41,976.1
FOME	54.5	182.7	6,061.4	33,544.5	59,301.9
Rail	152.5	212.9	N/A[2]	29,012.2	28,899.9
PDS	179.6	224.5	34,118.3	115,200.0	115,200.0
Grand Total	**421.8**	**688.4**	**40,966.9**	**215,387.9**	**258,719.8**

[1]Infeasible problem set solution time for CPLEX does not include CPLEX2.mps in summation.
[2]None of the Rail problems for GLPK could be solved due to read error.
[3]lp_solve included 1 Netlib time out, 1 Kennington time out, 1 FOME time out, 2 Rail time outs, and 8 PDS time outs.
[4]MINOS included 3 FOME time outs, 2 Rail time outs, and 8 PDS time outs, 8 solve Small COO solve errors and 8 Large OCC solve errors.

`https://prod-ng.sandia.gov/techlib-noauth/access-control.cgi/2013/138847.pdf`

Evolution de performance d'un solveur PL/PLNE

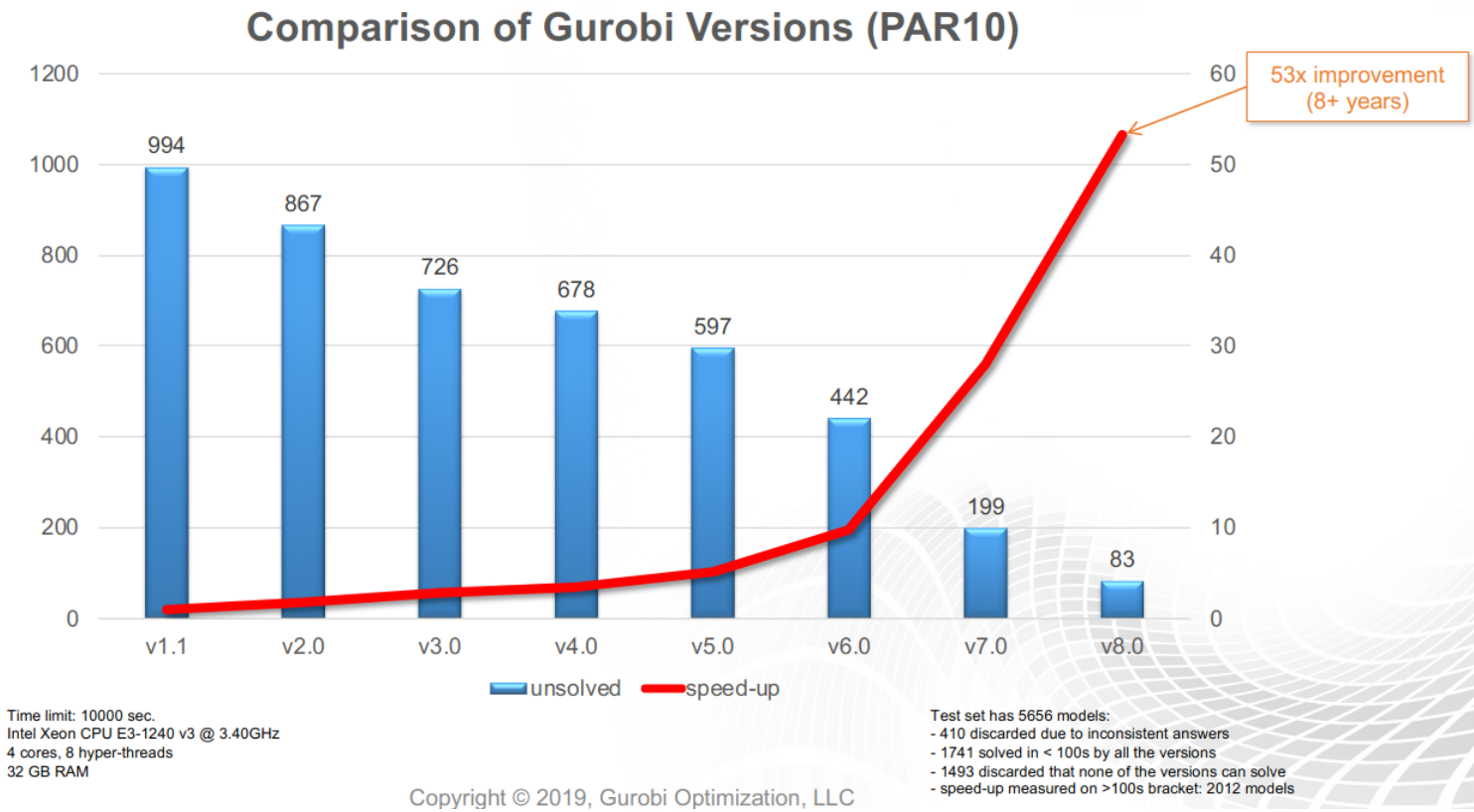
Broad Performance Improvements in v8.0

GUROBI OPTIMIZATION

- Consistent with prior releases, the Gurobi Optimizer v8.0 delivers performance improvements over v7.0 across a broad range of model types:
  - **MIP – 57%** faster overall, 109% faster on models that take >100 seconds to solve
  - **LP**
    - Concurrent – **15%** faster overall, 46% faster on models that take >100 seconds to solve
    - Primal Simplex – **24%** faster overall, 49% faster on models that take >100 seconds to solve
    - Dual Simplex – **32%** faster overall, 82% faster on models that take >100s to solve
    - Barrier – **13%** faster overall, 44% faster on models that take >100s to solve
  - **MIQP – 2.76x** faster overall
  - **MIQCP – 20%** faster overall
  - **SOCP – 19%** faster overall

`https://www.gurobi.com/pdfs/benchmarks.pdf`

Evolution de performance d'un solveur PL/PLNE



`https://www.gurobi.com/pdfs/benchmarks.pdf`

Que retenir de ces Benchmarks

- Les algorithmes de résolution PL implémentés dans les solveurs sont loin du simplexe datant de 1947.
- Les performances sont très différentes selon les solveurs, avec une concurrence acharnée (impact financier ou opérationnel gigantesque pour les utilisateurs clients).
- Les performances des solveurs sont constamment améliorées.
- Il ne sert à rien d'implémenter soi même un simplexe, ce sera bcp moins performant que les meilleurs solveurs libres et que les solveurs commerciaux.

$\Longrightarrow$ La résolution PL, il faut utiliser les solveurs existants tel quel, en "boîte noire".

$\Longrightarrow$ En résolution PLNE, il y aura un impact plus fort sur la manière d'utiliser les solveurs.

Plan

Rappel: Relaxation et bornes inférieures

Soit A, B dans $\mathbb{R}^n$ avec $A \subset B$, soit $f : B \longrightarrow \mathbb{R}$ une fonction bornée.

$\inf_{x \in A} f(x) = \inf\{f(x) | x \in A\}$, $\inf_{x \in B} f(x)$, $\sup_{x \in A} f(x) = \sup\{f(x) | x \in A\}$, $\sup_{x \in B} f(x)$ sont bien définis et:

$$\inf_{x \in A} f(x) \geqslant \inf_{x \in B} f(x) \tag{3}$$

$$\sup_{x \in A} f(x) \leqslant \sup_{x \in B} f(x) \tag{4}$$

(plus on a d'éléments à considérer, plus le minimum est bas)
(plus on a d'éléments à considérer, plus le maximum est élevé)

Dans le cas où B est bornée dans $\mathbb{R}^n$, f est continue, les hypothèses sont vérifiées

Si de plus B est fermé (alors compact), les bornes inférieures et supérieures sont des min et max: les bornes inférieures et supérieures sont atteintes

Rappel: relaxation continue

Pour un programme mixte en nombre entiers à minimiser:

$$\begin{array}{rl} & \min \sum_{i=1}^{p+m} c_i x_i = c.x \\ s.c: & A.x \geqslant b \\ & x \in \mathbb{N}^m \times \mathbb{R}_+^p \end{array} \tag{5}$$

La relaxation continue, donne une borne inférieure (duale):

$$\begin{array}{rl} & \min c.x \\ s.c: & A.x \geqslant b \\ & x \geqslant 0 \end{array} \leqslant \begin{array}{rl} & \min c.x \\ s.c: & A.x \geqslant b \\ & x \in \mathbb{N}^m \times \mathbb{R}_+^p \end{array} \tag{6}$$

Une solution réalisable x^0 vérifiant les contraintes $Ax^0 \geqslant b$ est un majorant de l'optimum (borne primale):

$$\begin{array}{rl} & \min c.x \\ s.c: & A.x \geqslant b \\ & x \in \mathbb{N}^m \times \mathbb{R}_+^p \end{array} \leqslant c.x^0 = \sum_{i=1}^{p+m} c_i x_i^0 \tag{7}$$

$\Longrightarrow$ Donne un encadrement de la qualité de solution optimale d'un PLNE.

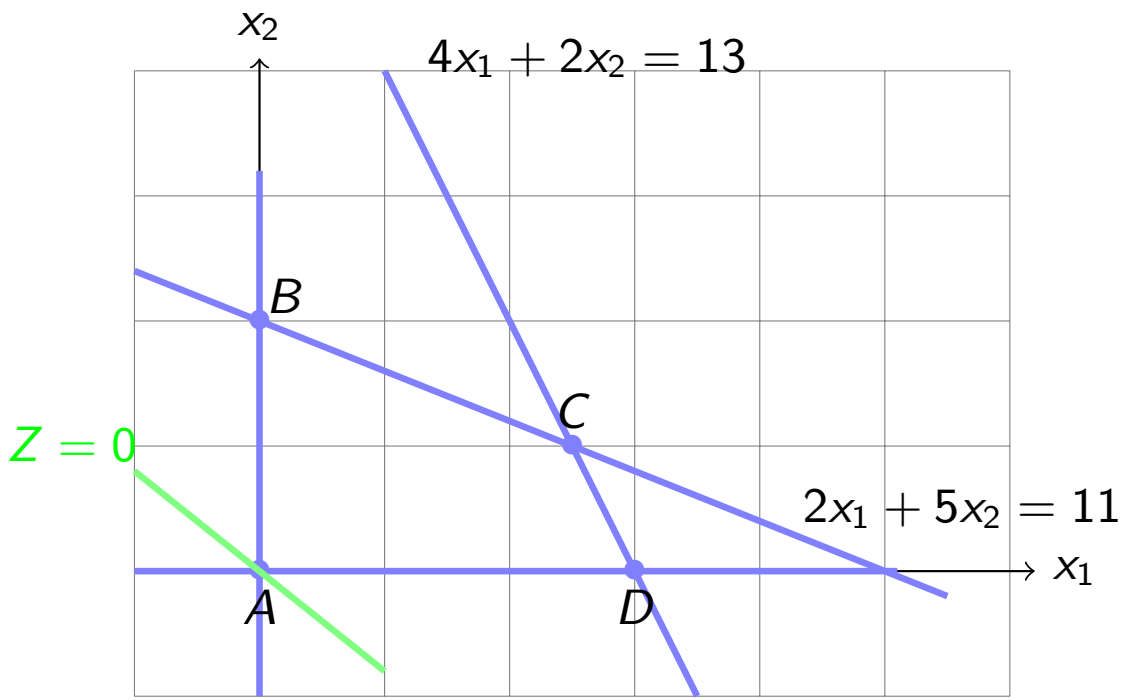
Cas où résolution PL donne des solutions entières

- La résolution PL (par algo polynomial) peut donner une solution entière.
- Dans ce cas, la solution est optimale dans le PLNE par encadrement.

$$\begin{cases} Z^* = \min 1200x_1 + 1500x_2 \\ 4x_1 + 2x_2 \leqslant 13 \\ 2x_1 + 5x_2 \leqslant 11 \\ x_1, x_2 \in \mathbb{N} \end{cases}$$

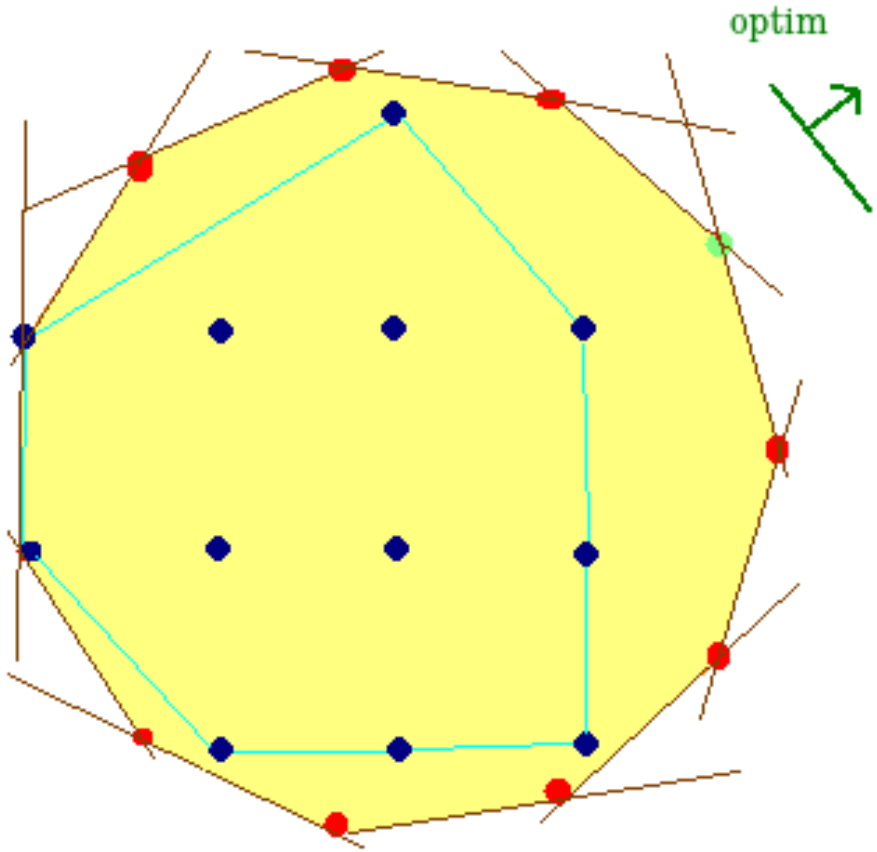
Solution optimale de relaxation continue: (0,0)

(0,0) est l'optimum du problème discret



- Un cadre idéal: lorsque tous les points extrêmes du polyèdre défini par relaxation continue sont entiers, la résolution du PLNE se ramène juste à un calcul de la relaxation PL pour toute fonction objectif.

Point de vue géométrique



$\Longrightarrow$ Si le polyèdre de la relaxation continue est collé au polyèdre défini par les solutions entières, la résolution PL donne une solution au PLNE.

$\Longrightarrow$ En général, la relaxation continue donne juste une borne duale.

Matrice totalement unimodulaire

Définition (Matrice totalement unimodulaire)

On dit qu'une matrice $M \in \mathbb{Z}^{m \times n}$ *est totalement unimodulaire si toute sous-matrice carrée extraite de* M *est de déterminant* 0 *ou* ± 1.

Lemme de Poincaré, une condition suffisante plus facile à vérifier

Proposition (Condition suffisante de totale unimodularité)

Soit une matrice $M \in \{-1, 0, 1\}^{m \times n}$ *avec au plus un coefficient* -1 *par colonne, et au plus un coefficient* 1 *par colonne. Alors* M *est totalement unimodulaire.*

Pour en savoir plus sur les matrices totalement unimodulaires:

L.A. Wolsey. Integer Programming, J. Wiley, 1998

G.L. Nemhauser and L.A.Wolsey. Integer and Combinatorial Optimization, J.Wiley, 1999

Matrice totalement unimodulaire et points extrêmes entiers

Théorème

Soit $M \in \mathbb{Z}^{m \times n}$ une matrice totalement unimodulaire, soient $l \in (\mathbb{Z} \cup \{-\infty\})^n$, $u \in (\mathbb{Z} \cup \{+\infty\})^n$, $a \in (\mathbb{Z} \cup \{-\infty\})^m$, $b \in (\mathbb{Z} \cup \{+\infty\})^m$.

Alors les points extrêmes du polyèdre suivant sont entiers:

$$\mathcal{P} = \left\{ x \in \mathbb{R}^n \mid a \leqslant M.x \leqslant b, \;\; l \leqslant x \leqslant u \right\}$$

En particulier, les problèmes d'optimisation $\min_{x \in \mathcal{P}}$ / $\max_{x \in \mathcal{P}}$ $c.x$ se résolvent en temps polynomial, un calcul PL fournit une solution optimale entière.

$\Longrightarrow$ outil pour prouver qu'un problème d'optimisation a une complexité polynomiale

Applications de totale unimodularité

En utilisant le lemme de Poincaré et le théorème précédent, on peut prouver que les problèmes d'optimisation suivants sont polynomiaux:

- flot-max (et min-cut) dans un graphe, on le savait déjà par l'algorithme de Ford Fulkerson-Karp
- Problème d'affectation simple, on le savait déjà par l'algorithme hongrois. (mentionné au CM4, avec la résolution Excel)
- Problème de plus court chemin avec poids positifs, on le savait déjà par l'algorithme de Dijkstra.

$\Longrightarrow$ Sur ces problèmes,il vaut mieux utiliser l'algorithme spécifique qu'une résolution PL.

$\Longrightarrow$ Il y a d'autres applications de la totale unimodularité (notamment pour la robustesse de Bertsimas et Sim)

Calculer l'enveloppe convexe?

- Si on peut calculer l'enveloppe convexe des points entiers ou une description linéaire de l'enveloppe convexe, il reste alors u calcul de PL pour trouver une solution optimale entière.
- C'est une telle méthode que l'on fait graphiquement en dimension 2.
- Problème: à partir de polyèdres défini par des contraintes $\mathcal{P} = \left\{x \in \mathbb{R}^n \mid a \leqslant M.x \leqslant b,\ \ l \leqslant x \leqslant u\right\}$, on n'a pas d'algorithme polynomial pour calculer l'enveloppe convexe des points entiers
- L'enveloppe convexe des points entiers peut être de taille exponentielle (c'est même souvent le cas).

Calculer l'optimum discret par itération de PLs?

- Idée: peut on raffiner la description polyédrale des contraintes uniquement autour de l'optimum entier pour prouver l'optimalité par un calcul PL?
- Coupes de Chvatal-Gomory: coupes d'arrondis, valables pour tout PLNE, permettent de couper une solution partielle obtenue par PL, et itérations convergent vers l'optimum entier.
- Difficulté: complexité exponentielle en général en convergeant en un nombre exponentiel d'itérations

$\Longrightarrow$ Méthode inefficace en pratique

Approches polyédrales

- On peut parfois avoir une description avec un nb exponentiel de contraintes de l'enveloppe convexe des points entiers.
- Démonstration: prouver que des contraintes définissent des facettes de l'enveloppe convexe.
- Dans certains cas , on peut résoudre un tel PL avec un nb exponentiel de contraintes en temps polynomial (on le verra au prochain cours).

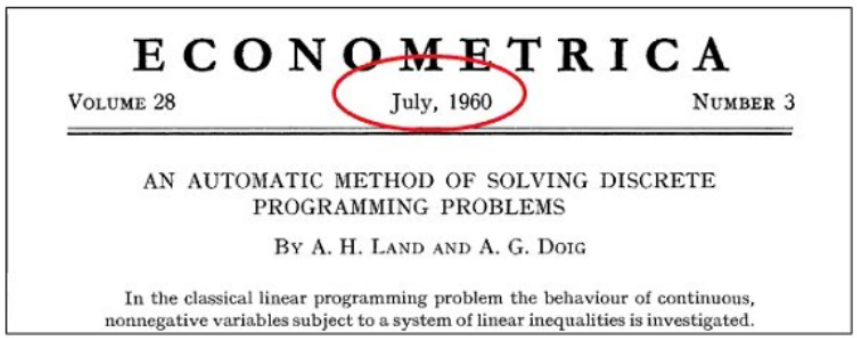
$\Longrightarrow$ Autre outil pour prouver que des problèmes d'optimisation (ou des sous-cas particuliers) sont polynomiaux

`https://www.lamsade.dauphine.fr/~mahjoub/MODO/Chapitre-Polyedres.pdf`
`http://ejc-gdr-ro.event.univ-lorraine.fr/docs/RM.pdf`

Plan

Fondatrices de l'algo de Branch&Bound

Ailsa Land



ECONOMETRICA

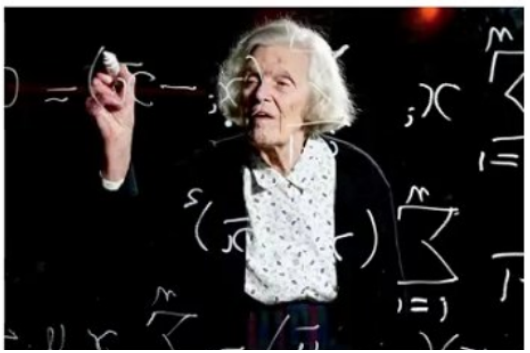
VOLUME 28 July, 1960 NUMBER 3

AN AUTOMATIC METHOD OF SOLVING DISCRETE PROGRAMMING PROBLEMS

BY A. H. LAND AND A. G. DOIG

In the classical linear programming problem the behaviour of continuous, nonnegative variables subject to a system of linear inequalities is investigated.

Alison Harcourt



Ailsa H. Land nous a quitté le 16 mai 2021, l'INFORMS lui a bien sûr rendu hommage: `https://www.informs.org/Explore/History-of-O.R.-Excellence/Biographical-Profiles/Land-Ailsa-H`

Algorithme générique de résolution d'un PLNE

Les opérateurs utilisés pour le problème de sac à dos sont génériques! Pour un problème de minimisation, les principes de l'algorithme de Branch&Bound (ou séparation-évaluation) guidés par la relaxation PL:

- La résolution continue, résolution PL, donne des bornes inférieures.
- élagage: un noeud est élagué (inutile à explorer) si sa borne inférieure est supérieure à (au moins aussi bonne que) la meilleure solution connue.

  N.B: en pratique, il peut y avoir une tolérance numérique, absolue (1 pour une fonction objectif entière) ou relative (par exemple, dans un intervalle de 0.01%)
- Branchements: Fixer des variables à certaines valeurs, ou ajouter des contraintes linéaires, préserve la structure PLNE, on peut utiliser les bornes PL à chaque noeud. Cela restreint les possibilités, et augmente la borne inférieure du noeud père.
- Stratégie générique pour le choix du noeud à traiter : aléatoire, BFS, DFS, BestBound, ou plus élaborée.

Règles de branchements en Branch and Bound

- Branchement: couper le problème en deux sous problèmes plus petit, selon un hyperplan
- Branchement sur une variable judicieusement choisie x, de valeur $\tilde{x}$ fractionnaire dans la solution duale. On distingue les cas en considérant deux noeuds en rajoutant les contraintes $x \geqslant \lceil \tilde{x} \rceil$ et $x \leqslant \lfloor \tilde{x} \rfloor$
- Branchement SOS, pour variables x_i tq $\sum_{i \in I} x_i \leqslant 1$. Pour une partition de I, $I_1 \cup I_2 = I$ et $I_1 \cap I_2 = \emptyset$ avec $|I_1| \approx |I_2| \approx \frac{|I|}{2}$. On branche selon les hyperplans $\sum_{i \in I_2} x_i = 0$ et $\sum_{i \in I_1} x_i = 0$
- Les branchements permettent de se rapprocher de l'enveloppe convexe des points entiers (cf illustration suivante).

Illustration du branchement et de la convexification

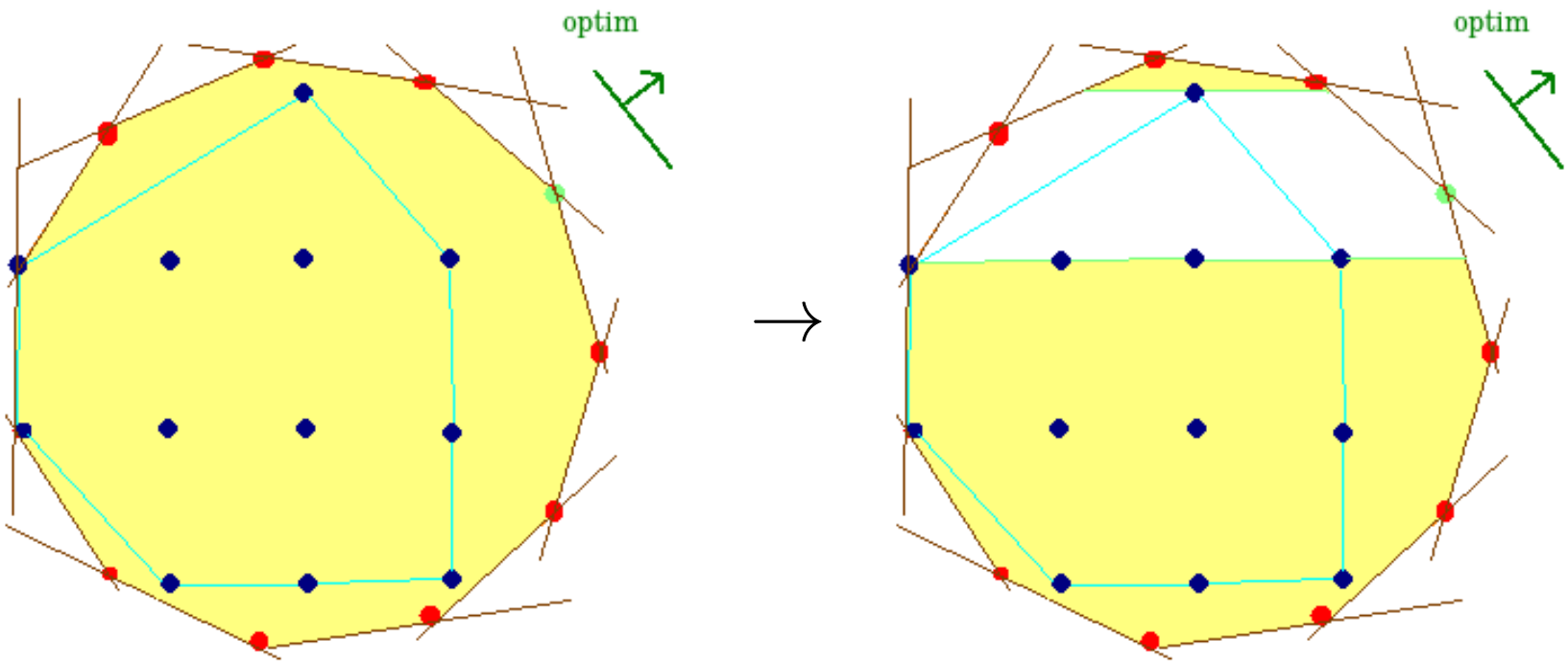
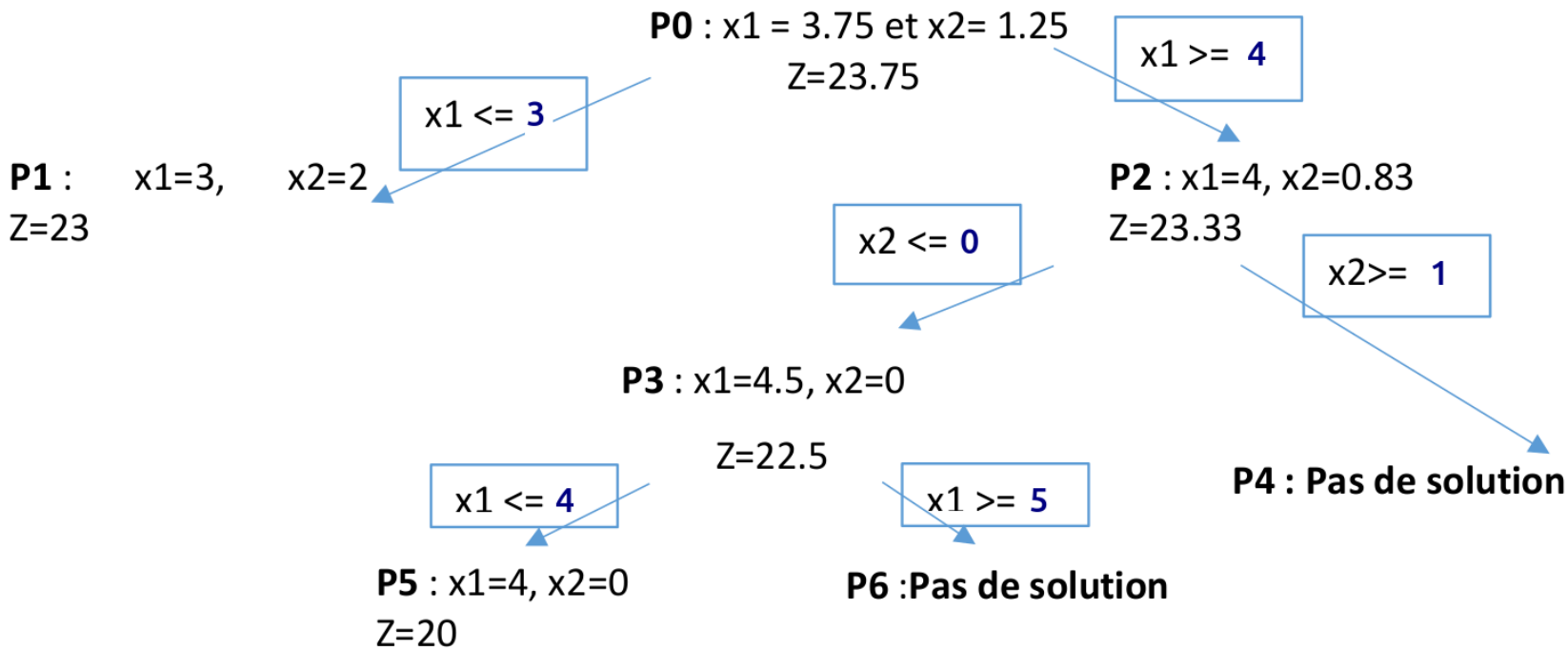


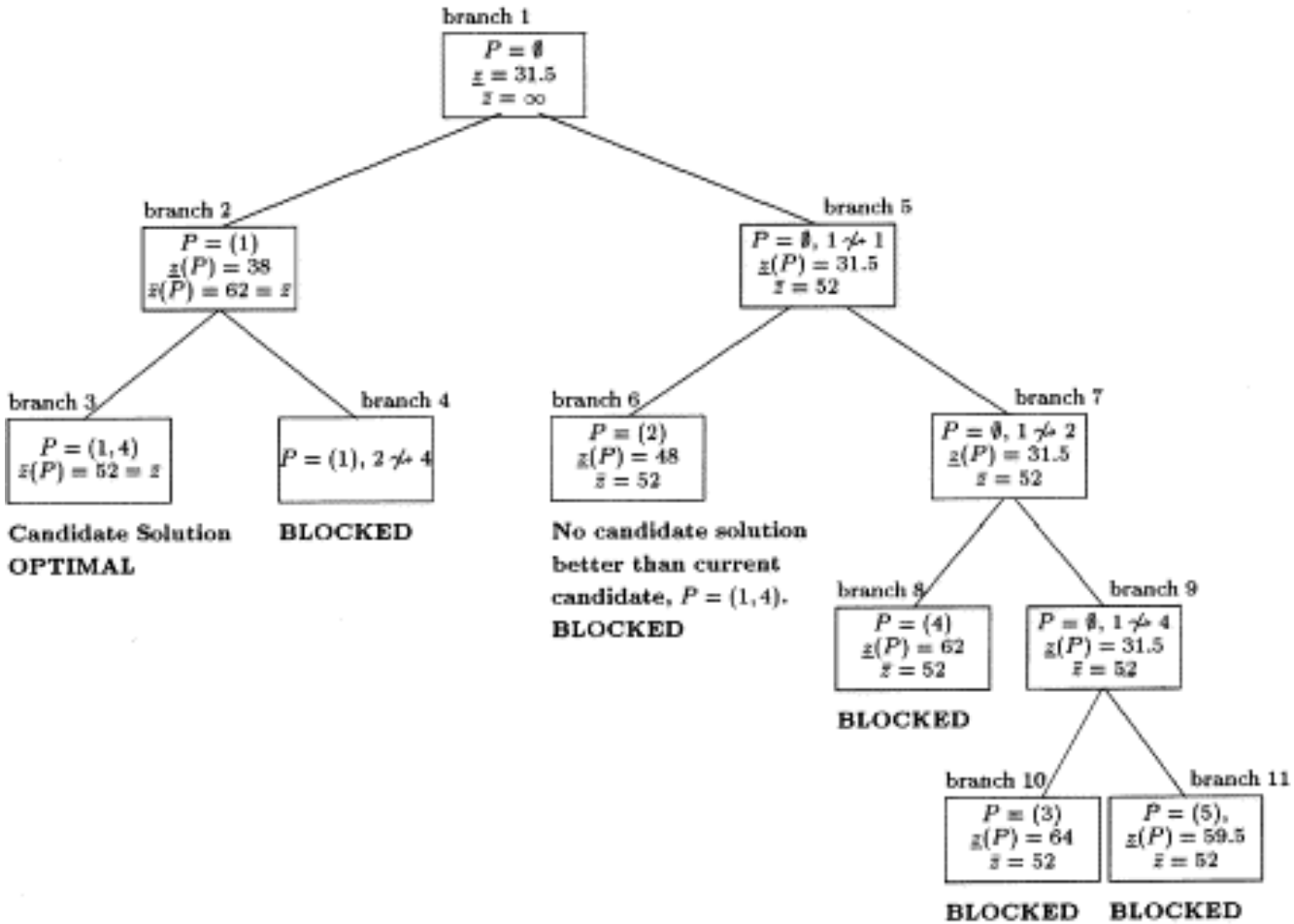
Illustration pour un pb de maximisation

Algo de Branch&Bound déployé pour maximiser $5x_1 + 4x_2$ sous contraintes $x_1 + x_2 \leqslant 5$, $10x_1 + 6x_2 \leqslant 45$, $x_1, x_2 \in \mathbb{N}$



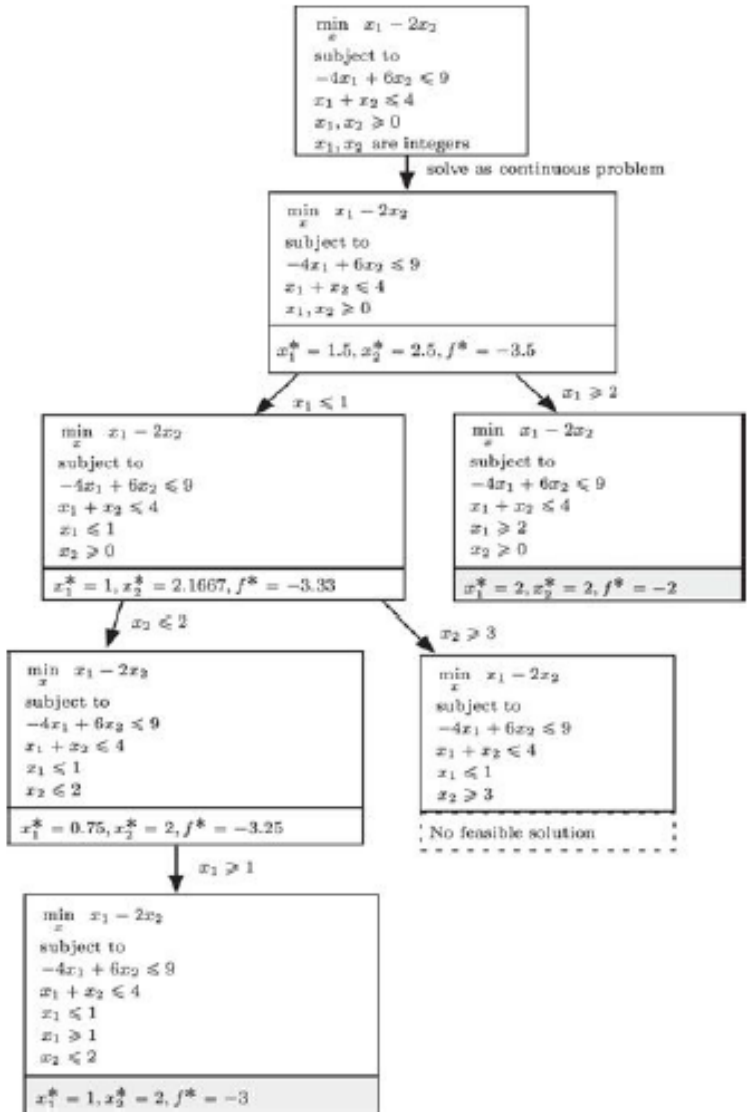
$\Longrightarrow$ La borne de relaxation continue est une borne supérieure, diminue au cours du temps et le le long de l'arbre

Illustration pour un pb de minimisation (1)



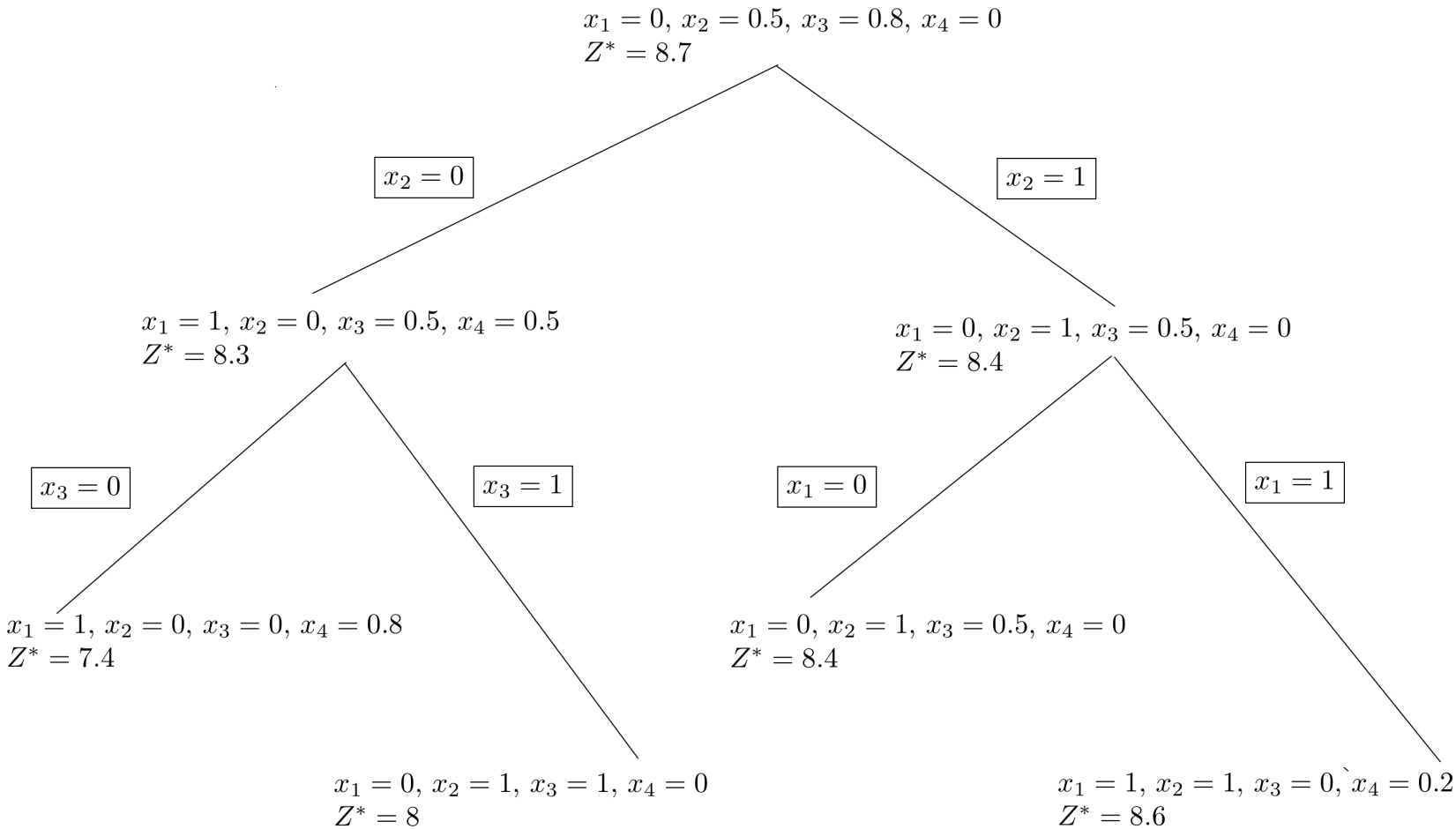
$\Longrightarrow$ La borne de relaxation continue est une borne inférieure, augmente au cours du temps et le long de l'arbre

Illustration pour un pb de minimisation (2)



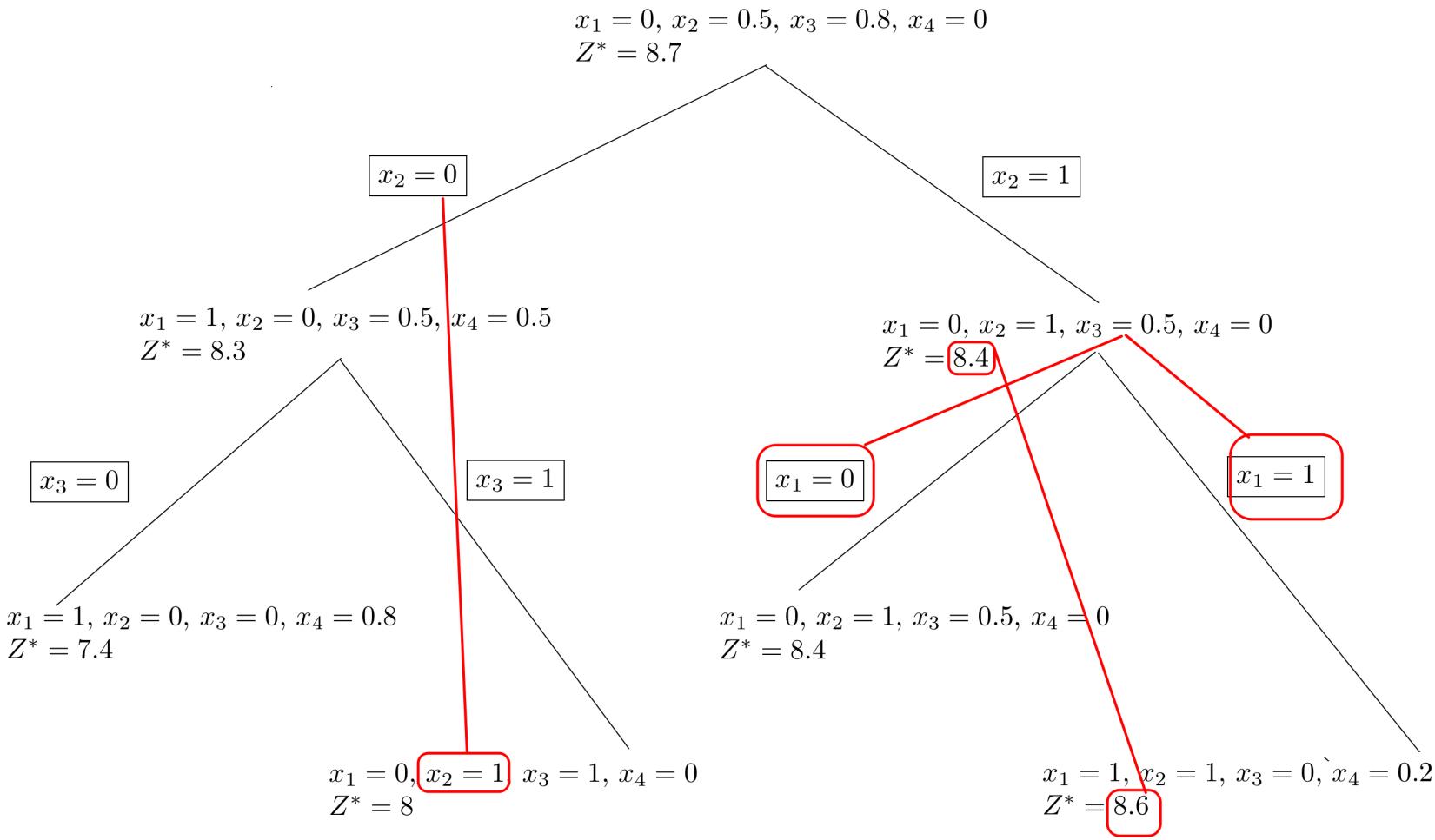
$\Longrightarrow$ La borne de relaxation continue est une borne inférieure, augmente au cours du temps et le long de l'arbre

Jeu des 3 erreurs



$\Longrightarrow$ Quelles sont les 3 erreurs manifestes?

Jeu des 3 erreurs, solutions



Rappel: Simplexe dual et ajout de contraintes

Avec la dualité forte et l'ajout de contraintes:

$$\begin{array}{lcllcll} \min b^T.y + b'^T.y' & = & \max c.x & \leqslant & \max c.x & = & \min b^T.y \\ A^T.y + A'^T.y' \geqslant c^T & & A.x \leqslant b & & A.x \leqslant b & & A^T.y \geqslant c^T \\ y, y' \geqslant 0 & & A'.x \leqslant b' & & x \geqslant 0 & & y \geqslant 0 \\ & & x \geqslant 0 & & & & \end{array}$$

A partir de la solution optimale initiale y dans le dual, en complétant avec $y' = 0$, cela forme une solution dual réalisable dans le problème avec contrainte additionnelle.

Très utile pour initialiser le simplexe dual, c'est un "démarrage à chaud", peu d'itérations seront nécessaires pour calculer l'optimum du problème avec la contrainte ajoutée.

$\Longrightarrow$ Dans l'algo de B&B, on calcule alors plus rapidement les relaxations continues après branchement, en partant de la solution continue du noeud initial, on ajoute juste une contrainte de branchement.

N.B: opérateur généralisant le "démarrage à chaud" de l'algo glouton continu

Une différence majeure avec le sac à dos

- Dans le cas d'un sac à dos, la relaxation continue avait au plus une variable fractionnaire, pas de choix sur quelle variable "brancher".
- Max-stable: toutes les valeurs peuvent être fractionnaires, et valoir 0.5 dans la relaxation continue: on a l'embarras du choix!
- Stratégies de branchement: quelle variable fractionnaire choisir pour créer deux noeuds de l'arbre en distinguant les cas sur la valeur de la fractionnaire choisie.

Achterberg, T., Koch, T., Martin, A.: Branching rules revisited. Operations Research Letters 33(1), 42–54 (2005)

Eléments clés pour l'efficacité pratique

Améliorer les bornes inférieures à un noeud, peut permettre de supprimer des noeuds plus tôt:

- Le travail de modélisation PLNE influe sur la qualité des relaxations continues.
- Ajout de coupes d'intégrité: contraintes valides sur les points entiers permettant de couper des solutions fractionnaires.

Avoir plus rapidement une bonne solution (primale) peut permettre de supprimer des noeuds plus tôt:

- Heuristiques génériques pour chercher des bonnes solutions à chaque noeud.
- Démarrage à chaud: fournir initialement une bonne solution (primale) accélère la résolution PLNE. Une heuristique spécialisée en amont de la PLNE par exemple.

De nombreuses stratégies de branchements, peuvent avoir une grande influence en pratique

En pratique, de nombreux paramètres et algorithmes heuristiques à l'intérieur de la résolution PLNE.

Symétries et résolution par algo de Branch&Bound

- Pour le pb de coloration des sommets d'un graphe, on avait vu que bcp de solutions symétriques peuvent être obtenues par permutation,
- Dans l'algorithme de B&B, une telle structure est catastrophique, cela induit de duppliquer en grand nombre des parties de l'arbre de B&B, avec les permutations.
- La borne duale s'améliore très lentement: des sous cas optimaux dans de multiples parties de l'arbre B&B
- Vous l'expérimenterez avec GLPK, ici intérêt à avoir une formulation avec moins symétries, comme la formulation par représentant.
- Une difficulté générale en PLNE. Mieux vaut casser soi-même des symétries, les algorithmes automatiques des solveurs sont très limités.

Rappels: Premiers éléments d'analyse polyédrale

Soit $\mathcal{P}$ un problème d'optimisation discret portant sur des variables $x \in \mathbb{N}^m$ avec $c.x$ à minimiser.

On suppose avoir deux descriptions linéaires du problème, $Ax \geqslant a$ et $Bx \geqslant b$ pour $x \geqslant 0$: $\{x \geqslant 0 |\ Ax \geqslant a\} \cap \mathbb{N}^m = \{x \geqslant 0 |\ Bx \geqslant b\} \cap \mathbb{N}^m$ définissent les solutions réalisables de $\mathcal{P}$.

La formulation $Ax \geqslant a$ est plus forte si elle contient moins de solutions continues: $\{x \geqslant 0 |\ Ax \geqslant a\} \subset \{x \geqslant 0 |\ Bx \geqslant b\}$

On a alors pour toute fonction objectif définie par des valeurs de c:
$\min_{x \geqslant 0, Ax \geqslant a} c.x \geqslant \min_{x \geqslant 0, Bx \geqslant b} c.x$ même si $\min_{x \in \mathbb{N}^m, Ax \geqslant a} c.x = \min_{x \in \mathbb{N}^m, Bx \geqslant b} c.x$

La relaxation continue est de meilleure qualité avec des contraintes plus fortes.

$\Longrightarrow$ Lien entre inclusions de polyèdres et qualité de relaxation continue

$\Longrightarrow$ Impact numérique en PLNE sur l'élagage en fonction des qualités de relaxation continue

Influence des coupes sur les bornes de relaxation continue

Instance Cuts	Optimum	LP relax No	LP relax 1 order cuts	LP relax 2 order cuts
data1.3.5.10.ex1	438	268	434	434
data1.3.5.10.ex2	513	303	452	465
data1.3.5.10.ex3	649	415	591	591
data1.3.5.10.ex4	449	130	391	393
data1.8.15.20.ex1	1000	677	899	922
data1.8.15.20.ex2	1229	744	1142	1165
data1.8.5.20.ex1	693	431	580	593
data1.8.5.20.ex2	636	283	475	489
data1.8.5.20.ex3	712	301	636	662
data1.8.5.20.ex4	868	295	631	650
data1.8.5.20.ex5	675	317	539	555
data1.15.20.40ex1	1462	617	1239	1243
data1.15.20.40ex2	1422	699	1280	1318
data1.15.20.40ex3	1515	634	1266	1276
data1.15.20.40ex4	1386	630	1215	1236
Total Gap	15427	7759 49,7%	13478 12,6%	13748 10,9%

Instances faciles d'un problème de type VRPTW, influence des coupes de sous tour d'ordres 1 et 2

N. Dupin, R. Parize, E-G Talbi, Matheuristics and Column Generation for a Basic Technician Routing Problem. Algorithms, 14(11), 313, 2021.

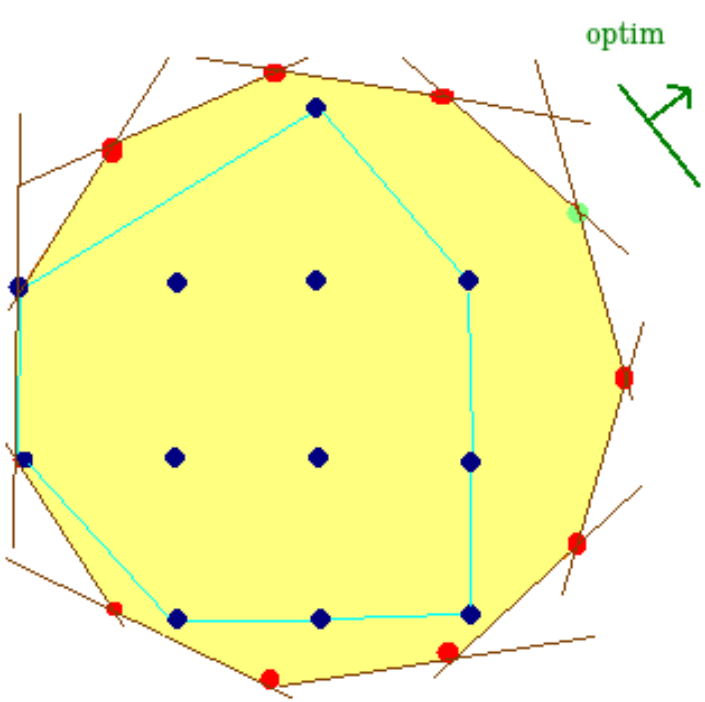
Influence des coupes de sous tour entre deux villes sur les temps de calcul

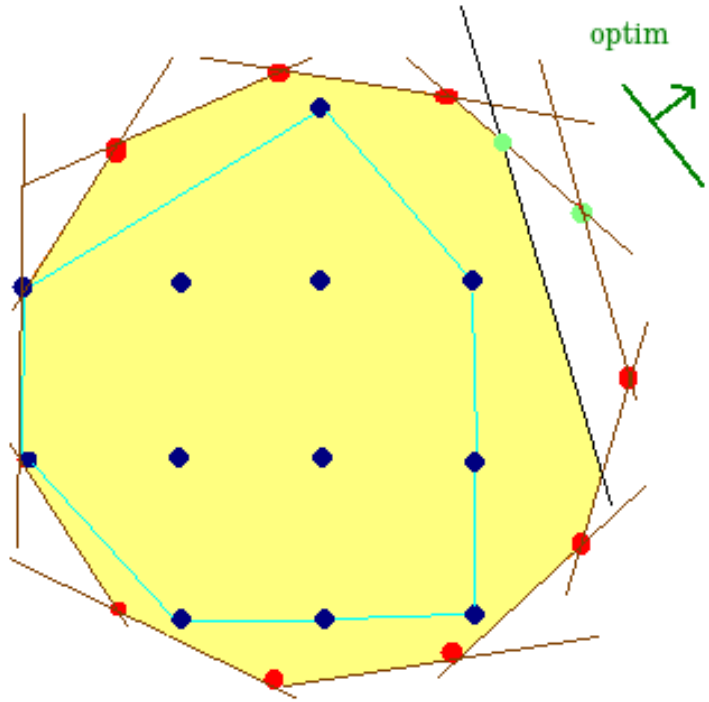
Solveur Coupes?	GLPK Sans	GLPK Avec	Cplex Sans	Cplex Avec
Dat.8.15.20.ex1	17,5	1,9	1,75	0,04
Dat.8.15.20.ex2	4,7	0,8	0,41	0,05
Dat.8.5.20.ex1	8630,8	84,5	155,73	0,59
Dat.8.5.20.ex2	10128,4	593,8	350,78	13,64
Dat.8.5.20.ex3	5,8	1,9	0,5	0,13
Dat.8.5.20.ex4	43138,8	791,4	342,39	8,99
Dat.8.5.20.ex5	14631,2	149,7	32,94	0,6
Dat.15.20.40ex1	Mem error	13418,8	3347,05	34,97
Dat.15.20.40ex2	Mem error	236,9	39,08	0,92
Dat.15.20.40ex3	Mem error	1759,2	84,52	3,19
Dat.15.20.40ex4	Mem error	4739,8	16662,59	11,54
Bilan		21779,2	21020,1	74,91

Intel Core2 Duo processor, 2.80GHz.

GLPK et Cplex, dans leur version de 2010.

Illustration des coupes d'intégrité





$\Longrightarrow$ Eliminer par des coupes valides des points continus valides pour les contraintes initiales, améliore la qualité de la borne continue

$\Longrightarrow$ Idéalement, une coupe définit une facette du polyèdre défini par l'enveloppe convexe des points entiers

Coupes d'intégrité sur l'exemple déroulé

$$\min \quad x_1 + 1,5x_2 + 2x_3 + 5x_4 + 9x_5$$
$$s.c. \quad x_1 + 2x_2 + 3x_3 + 6x_4 + 10x_5 \geqslant 10$$
$$\forall i, \quad x_i \in \{0, 1\}$$

Solution continue : $x_2 = x_3 = 1,\ x_1 = x_5 = 0,\ x_4 = \frac{5}{6}$
Relâché continu : $1,5 + 2 + \frac{25}{6} \approx 7,67$

$$x_4 + x_5 \geqslant 1$$

Solution continue : $x_3 = x_4 = 1,\ x_2 = \frac{1}{2},\ x_1 = x_5 = 0$
Relâché continu : $5 + 2 + 0,75 = 7,75$

$\Longrightarrow$ La coupe aurait permis d'énumérer un cas en moins, on retrouvait directement un noeud visité.

Formalisation de manière générale

Partant d'un programme mixte en nombre entiers:

$$\begin{aligned} & \min \sum_{i=1}^{m+p} c_i x_i \\ s.c: \quad & \sum_{i=1}^{m+p} A_{i,c} x_i \geqslant b_c \quad \forall c \in [\![1; C]\!] \\ & x \in \mathbb{N}^m \times \mathbb{R}^p \end{aligned}$$

L'équation $(*) : \sum_{i=1}^{m+p} a_i x_i \geqslant d$ est une coupe valide si l'ajout de $(*)$ ne change pas la faisabilité des solutions entières, et potentiellement élimine des solutions continues ($x \in \mathbb{R}^{m+p}$) vérifiant $\sum_{i=1}^{m+p} A_{i,c} x_i \geqslant b_c$

En général, un grand nombre de coupes (indexation sur des ensembles de taille exponentielle), on ne génère que celles qui ont un impact sur la solution de la relaxation continue: améliore la borne inférieure au noeud considéré.

Algorithme de séparation: algorithme choisissant des coupes à ajouter parmi un grand ensemble de coupes, choix guidé par la relaxation continue.

Quelques types de coupes d'intégrité

- Coupes d'arrondis, génériques : coupes de Chvatal-Gomory, coupes "Zero half cuts"
- Coupes pour des contraintes de sac à dos: coupes de couvertures, couvertures GUB
- Coupes d'incompatibilités (graphe de dépendance) : Coupes de cliques, coupes de cycles
- Et bien d'autres types de coupes activées suivant la détection de structures spécifiques:

`https://www.ibm.com/support/knowledgecenter/SSSA5P_12.7.1/ilog.odms.cplex.help/CPLEX/UsrMan/topics/discr_optim/mip/cuts/26_cuts_title_synopsis.html`

Coupes d'arrondis (1)

ex: A partir d'une contrainte $1.5x_1 + 1.7x_2 + 1.8x_3 \leqslant 2.2$ où $x_1, x_2, x_3 \in \{0, 1\}$

On divise par 1.5: $x_1 + 1.7/1.5x_2 + 1.8/1.5x_3 \leqslant 2.2/1.5$

On minore à gauche par les parties entières
$\lfloor x_1 + 1.7/1.5x_2 + 1.8/1.5x_3 \rfloor \leqslant x_1 + 1.7/1.5x_2 + 1.8/1.5x_3 \leqslant 2.2/1.5$

Or :
$\lfloor x_1 + 1.7/1.5x_2 + 1.8/1.5x_3 \rfloor \geqslant x_1 + \lfloor 1.7/1.5x_2 \rfloor + \lfloor 1.8/1.5x_3 \rfloor = x_1 + x_2 + x_3$

On a alors: $x_1 + x_2 + x_3 \leqslant 2.2/1.5$

Le membre de gauche étant entier , on a $x_1 + x_2 + x_3 \leqslant 1 = \lfloor 2.2/1.5 \rfloor$

On obtient une contrainte qui définit les mêmes points entiers, "plus forte": si $x_1, x_2, x_2 + 3 \in [0, 1]^3$, $x_1 + x_2 + x_3 \leqslant 1 \Longrightarrow 1.5x_1 + 1.7x_2 + 1.8x_3 \leqslant 2.2$

$(1, 0.7/1.7, 0)$ était solution continue valide de la formulation originale, non valide avec $x_1 + x_2 + x_3 \leqslant 1$.

Coupes d'arrondis (2)

- Coupes de Chvatal-Gomory s'appliquent de manière générale pour des solutions fractionnaires d'une relaxation PL.
- En pratique, rares solutions factionnaires dans une relaxation PL, guident les cas où les arrondis et parties entières donnent des inégalités strictes.
- Résultats théoriques de convergence en utilisant uniquement de telles coupes, mais lent en pratique.
- Coupes zero half-cuts: on peut utiliser le procédé d'arrondi en sommant deux contraintes.

Coupes de couvertures

Les coupes de couvertures s'appliquent à des contraintes de sac à dos $\sum_i a_i x_i \leqslant d, a_i \geqslant 0$.

Une couverture: un ensemble C tq $\sum_{i \in C} a_i > d$.

On alors la contrainte:

$$\sum_{i \in C} a_i > d \Longrightarrow \sum_{i \in C} x_i \leqslant |C| - 1$$

Pour des contraintes $\sum_i a_i x_i \geqslant d, a_i \geqslant 0$, $x_i \leftrightarrow 1 - x_i$ on a :

$$\sum_{i \in C} a_i < d \Longrightarrow \sum_{i \notin C} x_i \geqslant 1$$

Rq: là encore, la relaxation continue guide le choix de la couverture C à utiliser. Une couverture candidate est définie par l'ensemble des variables strictement positives.

Coupes de couvertures GUB

Contraintes GUB $\sum_i x_i \leqslant 1$ peuvent renforcer des coupes de couverture.

Example: $x_1 + x_2 \leqslant 1$, $8x_1 + 7x_2 + 3x_3 + 4x_4 + 5x_5 \geqslant 12$

Coupe de couverture avec (x_1, x_3): $x_2 + x_4 + x_5 \geqslant 1$

(x_1, x_3) couverture $\Longrightarrow$ (x_2, x_3) couverture, donc $x_1 + x_4 + x_5 \geqslant 1$

Coupe de couverture GUB: un élément de contrainte GUB $x_1 + x_2 \leqslant 1$, et x_3 font une couverture.

la coupe de couverture renforcée est $x_4 + x_5 \geqslant 1$

Preuve: soit $z = x_1 + x_2$. On a $8z + 3x_3 + 4x_4 + 5x_5 \geqslant 12$
(z, x_3) est une couverture, la coupe de couverture est $x_4 + x_5 \geqslant 1$

Graphe de dépendance

En preprocessing de résolution B&B, un graphe de dépendance est construit:

Sommets: les variables binaires x et leurs complémentaires $\bar{x} = 1 - x$ du PLNE.

Arêtes: 2 sont reliés si les variables associées sont incompatibles.

Par exemple:
$x_1 + x_2 \leqslant 1$, incompatibilité entre x_1 et x_2;
$x_1 \leqslant x_2 \Longrightarrow x_1 + 1 - x_2 \leqslant 1 \Longrightarrow x_1 + \bar{x_2} \leqslant 1$, incompatibilité entre x_1 et $\bar{x_2}$

De manière générale, lorsque des implications suivantes sont détectées:
$x = 1 \Longrightarrow y = 1$ se traduit par $x \leqslant y$, x et $\bar{y}$ incompatibles
$x = 1 \Longrightarrow y = 0$ se traduit par $x + y \leqslant 1$, x et y incompatibles
$x = 0 \Longrightarrow y = 1$ se traduit par $1 \leqslant x + y$, $\bar{x}$ et $\bar{y}$ incompatibles
$x = 0 \Longrightarrow y = 0$ se traduit par $y \leqslant x$, y et $\bar{x}$ incompatibles

Savelsbergh, M.W.: Preprocessing and probing techniques for mixed integer programming problems. ORSA Journal on Computing 6(4), 445–454 (1994)

Coupes de cliques et de cycle

Dans le graphe de dépendance, une clique $\mathcal{C} = \left((x_i)_{i\in I}, (\bar{x}_j)_{j\in J}\right)$ induit la coupe suivante:

$$\sum_{x\in\mathcal{C}} x \leqslant 1 \Longleftrightarrow \sum_{i\in I} x_i + \sum_{j\in J}(1-x_j) \leqslant 1$$

Preuve: par raisonnement de clique ou en sommant les inégalités par deux variables et processus d'arrondi

Remarque: énumérer toutes les cliques d'un graphe est NP-complet, on génère juste les coupes de cliques intéressantes par rapport aux solutions fractionnaires

Dans le graphe de dépendance, un cycle $\mathcal{C} = \left((x_i)_{i\in I}, (\bar{x}_j)_{j\in J}\right)$ induit la coupe suivante:

$$\sum_{x\in\mathcal{C}} x \leqslant 1 \Longleftrightarrow \sum_{i\in I} x_i + \sum_{j\in J}(1-x_j) \leqslant \left\lfloor \frac{|C|}{2} \right\rfloor$$

Remarque: idem, pas d'énumération totale des cycles. Coupes de cycles moins fortes que coupes de cliques en général, coupes de cycles utiles pour cycles de cardinaux impairs.

Savelsbergh, M.W.: Preprocessing and probing techniques for mixed integer programming problems. ORSA Journal on Computing 6(4), 445–454 (1994)

Exemple de coupe de Clique

max $1.1x_1 + 1.2x_2 + 1.3x_3$

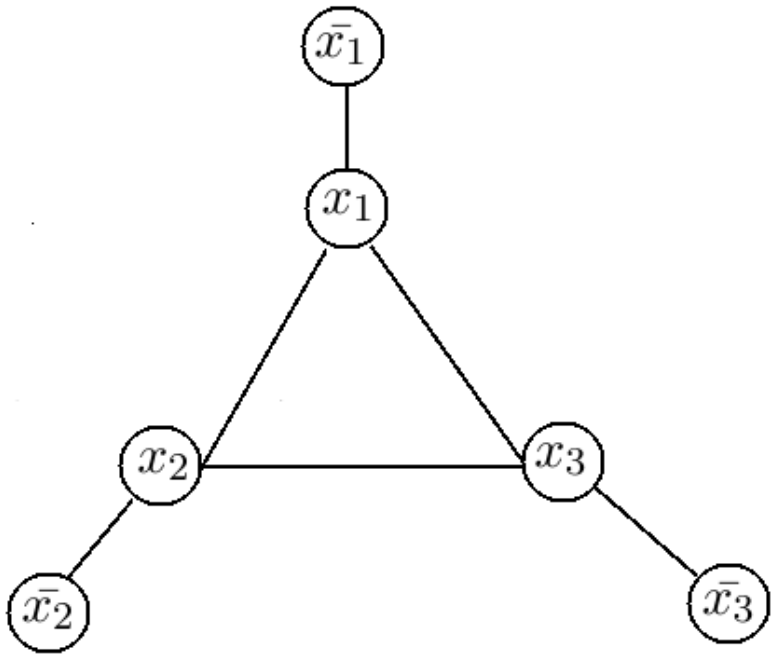
s.t:

$x_1 + x_2 \leqslant 1$

$x_2 + x_3 \leqslant 1$

$x_1 + x_3 \leqslant 1$

$x_1, x_2, x_3 \in \{0, 1\}$



Coupe de clique: $x_1 + x_2 + x_3 \leqslant 1$

Remarque: ce petit exemple s'obtient par procédé d'arrondi également, les contraintes sommées donnent $2x_1 + 2x_2 + 2x_3 \leqslant 3$, puis $x_1 + x_2 + x_3 \leqslant 1.5$ et $x_1 + x_2 + x_3 \leqslant 1$

Exemple de coupe de Clique (2)

max $1.1x_1 + 1.2x_2 + 1.3x_3 + 1.4x_4$

s.t:

$x_1 + x_2 \leqslant 1$

$x_2 + x_3 \leqslant 1$

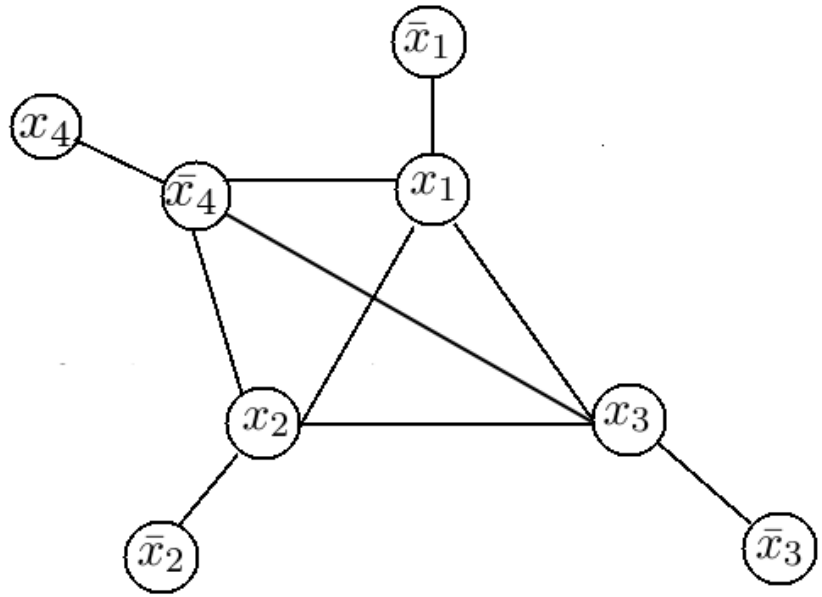
$x_1 + x_3 \leqslant 1$

$x_1 \leqslant x_4$

$x_2 \leqslant x_4$

$x_3 \leqslant x_4$

$x_1, x_2, x_3, x_4 \in \{0, 1\}$



Coupe de clique: $x_1 + x_2 + x_3 + (1 - x_4) \leqslant 1$ cad $x_1 + x_2 + x_3 \leqslant x_4$

Algorithme de séparation pour coupes de Clique

Rappel: il n'est pas envisageable d'énumérer toutes les cliques maximales du graphe de dépendance, pour déterminer lesquelles ajouter: celles qui sont violées dans la solution continue courante.

Question de séparation: existe-t'il une clique du graphe de dépendance $G = (V, E)$, dont la coupe de clique correspondante n'est pas réalisée dans la relaxation continue?

Autrement dit, étant donnée les n variables x_n du problème initial, de valeur $\tilde{x}_n$ dans la relaxation continue, y-a t'il une clique I, J du graphe de dépendance, telle que $\sum_{i \in I} \tilde{x}_i + \sum_{j \in J} (1 - \tilde{x}_j) > 1$?

Version optimisation: quel est le maximum de $\sum_{i \in I} \tilde{x}_i + \sum_{j \in J} (1 - \tilde{x}_j)$ lorsque I, J est une clique du graphe de dépendance?

En effet, l'existence d'une coupe de clique violée est équivalent à avoir un tel maximum de valeur > 1.

Question de séparation, formulation PLNE

Variables: $y_n \in \{0,1\}$ (resp $z_n \in \{0,1\}$) avec $y_n = 1$ (resp $z_n = 1$) si la variable initiale x_n (resp $\bar{x}_n = 1 - x_n$) est dans le graphe de dépendance

On note $V = X \cup X'$ avec $X = \{x_n\}$ et $X' = \{\bar{x}_n\}$, et respectivement $\bar{E}_{X-X}$, $\bar{E}_{X'-X'}$ et $\bar{E}_{X-X'}$ les couples de sommets non reliés dans E appartenant respectivement tous deux dans X et X' puis entre un sommet de X et un sommet de X'. On a la formulation PLNE suivante:

$$\max_{y,z \in \{0,1\}^n} \sum_{i=1}^{n} \tilde{x}_i y_i + \sum_{i=1}^{n} (1 - \tilde{x}_i) z_i$$

$$\begin{array}{lll} s.c: & \forall (i,j) \in \bar{E}_{X-X}, & y_i + y_j \leqslant 1 \\ & \forall (i,j) \in \bar{E}_{X'-X'}, & z_i + z_j \leqslant 1 \\ & \forall (i,j) \in \bar{E}_{X-X'}, & y_i + z_j \leqslant 1 \\ & \forall i \in [\![1;n]\!] & y_i, z_i \in \{0,1\} \end{array} \tag{8}$$

Si la solution optimale est > 1, on peut ajouter la contrainte $\sum_{i=1}^{n} y_i^* x_i + \sum_{i=1}^{n} z_i^* (1 - x_i) \leqslant 1$ où y^*, z^* est une solution optimale.

La séparation comme problème d'optimisation

Le problème de séparation, c'est résoudre un problème de clique max pondérée, un problème NP-complet.

Si on a résoudre comme problème initial clique max, trouver une coupe, ce serait résoudre un problème de clique max pondérée de même taille ...

On ne résout pas ce problème de clique max pondérée par une méthode exacte, il suffit de trouver une clique dont le coût est > 1, une heuristique rapide peut très bien faire l'affaire.

L'intérêt des coupes d'intégrité est d'améliorer la relaxation continue, sans obligation d'en ajouter. Il faut que le temps passé à la génération de coupes permette une accélération globale de l'algo B&B, que le temps gagné à énumérer moins de noeuds ne soit pas perdu par la génération de coupes et par des temps de calculs trop longs à chaque noeud.

En pratique, les solveurs ont des critères d'arrêt pour limiter les passes de générations de coupes, notamment si l'amélioration de relaxation continue devient peu significative.

Impact de coupes de sous tour et des coupes génériques

Impact des coupes génériques de Cplex pour améliorer la relaxation continue avant la phase de branchements pour différentes formulations compactes:

Bound Cuts	(1)–(8) LP No	(1)–(8) nod1 No	(1)–(8) LP (10)	(1)–(8) nod1 (10)	(12)–(19) LP No	(12)–(19) nod1 No	(12)–(19) LP (21)	(12)–(19) nod1 (21)
data_3_1_10	45.7%	8.7%	11.9%	3.2%	28.1%	3.3%	8.8%	1.0%
data_3_5_10	43.4%	0.0%	9.7%	0.0%	23.4%	0.0%	8.0%	0.0%
data_5_1_20	44.8%	29.5%	27.9%	11.6%	35.7%	17.9%	22.0%	13.7%
data_5_3_20	52.7%	30.3%	30.8%	17.7%	39.0%	19.9%	22.7%	15.5%
data_5_5_20	58.1%	28.7%	27.5%	14.6%	42.1%	18.5%	25.1%	14.1%
data_8_1_20	40.5%	20.6%	20.8%	10.6%	32.6%	10.6%	16.8%	7.8%
data_8_3_20	54.7%	28.5%	28.6%	17.1%	35.2%	14.8%	20.0%	11.2%
data_8_5_20	60.6%	26.7%	25.9%	11.2%	39.7%	8.1%	21.3%	1.6%
data_10_10_40	55.9%	25.0%	25.8%	14.3%	40.8%	19.4%	20.7%	13.9%
data_15_10_40	55.6%	18.9%	19.6%	10.7%	35.8%	11.0%	13.4%	6.3%
data_15_20_40	53.9%	14.2%	16.1%	6.1%	32.1%	7.5%	13.1%	4.8%

Les coupes génériques de Cplex à la racine (colonnes nod1), ne sont pas redondantes avec les coupes de sous-tour (notées cuts)

N. Dupin, R. Parize, E-G Talbi, Matheuristics and Column Generation for a Basic Technician Routing Problem. Algorithms, 14(11), 313, 2021.

Heuristiques primales en Branch& Bound

- Plongements (Diving heuristics): régulièrement, on effectue les branchements depuis un même noeud (plongement dans l'arbre B&B), jusqu' à trouver une solution réalisable ou infaisable. (DFS ne fait que ça)
- Heuristiques constructives: pour avoir une première solution réalisable au noeud racine (avant tout branchement), on peut utiliser la relaxation PL pour trouver une solution entière "proche". Ex: Feasibility Pump, RENS
- Heuristiques perturbatives: utilisent une solution réalisable pour en trouver de meilleures, ex RINS (Relaxation Induced Neighborhoods Search), Local Branching, VND-local Branching.
- Si un calcul PLNE s'achève, des heuristiques spécifiques pour "polir" la meilleure solution, utilisant différentes bonnes solutions réalisables

N.B: avec la difficulté de construction d'une première solution, il y a un intérêt majeur à fournir aux solveurs de PLNE une solution réalisable assez bonne, en utilisant une heuristique/méta-heuristique spécialisée pour problème, démarrage à chaud ("warmstart")

Références

• Berthold, T. RENS. Mathematical Programming Computation, 6(1), 33-54. (2014).

• Berthold, T.: *Primal Heuristics for Mixed Integer Programs*. Master thesis, Berlin, (2006).

• L. Bertacco, M. Fischetti, and A. Lodi. A feasibility pump heuristic for general mixed-integer problems. Discrete Optimization, 4(1):63–76, 2007.

• E. Danna, E. Rothberg, and C. Le Pape. Exploring relaxation induced neighborhoods to improve MIP solutions. Mathematical Programming, 102:71–90, 2005.

• M. Fischetti and A. Lodi. Local branching. Mathematical Programming, 98(1-3):23–47, 2003.

• Hansen, P., Mladenović, N., Urošević, D. Variable neighborhood search and local branching. Computers & Operations Research, 33(10), 3034-3045. (2006).

• Rothberg, E.: *An evolutionary algorithm for polishing mixed integer programming solutions* INFORMS Journal on Computing, vol 19, n 4, pp 534–541, 2007.

Heuristiques d'arrondis de solution continue?

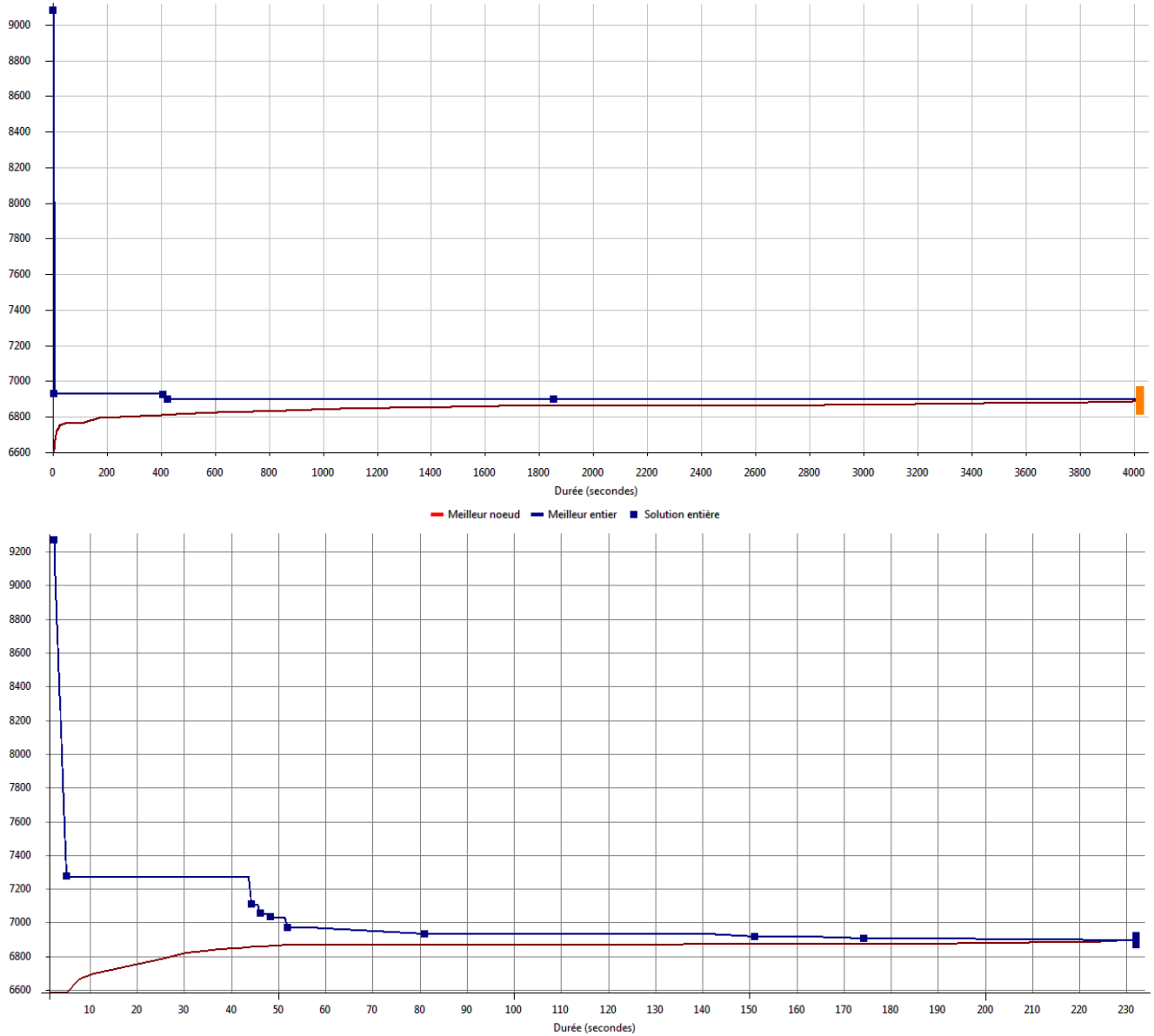
- A l'amphi graphes et PLNE, Vertex cover a une 2-approximation en arrondissant toutes les valeurs fractionnaires du PL.
- Heuristiques d'arrondis: résoudre une relaxation continue, fixer les variables entières et itérer arrondis et fixation impliquées.
- Fixer les variables entières dans relaxation continue et résoudre à l'optimalité (petit PLNE) définit une heuristique en général très rapide, mais de très mauvaise qualité en général. (N.B: c'est toujours meilleur que les heuristiques d'arrondis du point précédent)
- Feasibility Pump: chercher une solution entière proche de la relaxation continue. Pas super efficace en général, meilleure que fixation entière, a le mérite d'être une stratégie générique
- Des heuristiques de fixation de variables (éventuellement partielles) peuvent être efficaces, en utilisant des structures connues du problèmes, de l'expertise sur le problème ou de l'apprentissage:

N. Dupin, E-G Talbi, Parallel matheuristics for the discrete unit commitment problem with min-stop ramping constraints. International Transactions in Operational Research, Jan 2020, 27(1), pp 219-244.
F. Peschiera, R. Dell, J. Royset, A. Haït, N. Dupin, O. Battaïa, A novel solution approach with ML-based pseudo-cuts for the Flight and Maintenance Planning problem. OR Spectrum, 43, 2021, pp 635-664.

Heuristiques perturbatives RINS et Local Branching

- Un intérêt majeur de Feasibility Pump/RENS ou d'une heuristique externe: construire une solution réalisable même d'assez mauvaise qualité. (pb NP-difficile)
- La qualité des solutions primales est bien améliorée lorsqu'on a une solution réalisable, cela active des heuristiques génériques nécessitant une solution initiale
- RINS: Fixer les variables entières dans relaxation continue communes avec la meilleure solution primale et résoudre à l'optimalité (petit PLNE): définit une heuristique en général très rapide (peut être en temps tronqué), et de très bonne qualité en général. (la meilleure solution courante est solution de ce PLNE, peut même fournir un démarrage à chaud)
- Local Branching/VND Local Branching: utilisent uniquement la meilleure solution courante, systèmes de voisinages et de descente génériques en PLNE.

Convergence B&B avec modélisations PLNE différentes



N. Dupin, Tighter MIP formulations for the discretised unit commitment problem with min-stop ramping constraints, EURO Journal of Computational Optimization, 2017.

Primal bounds with MIP primal heuristics at the root node before branching

	Lv2	Lv2b	Lv3	Lv3b	St2	St2b	St3	St3b	Mx3
D6	0,31	0,06	**0,00**	0,06	0,04	0,04	**0,00**	0,04	**0,00**
D13	1,40	3,41	**1,97**	2,10	2,51	2,31	2,45	3,87	2,47
D16	0,17	0,16	0,89	0,39	0,07	0,14	0,09	0,05	**0,00**
D35	0,31	0,06	**0,00**	0,06	0,04	0,04	**0,00**	0,04	**0,00**
D48	3,12	**2,44**	4,71	3,10	5,67	4,65	3,33	4,63	3,20
D67	3,68	5,25	4,08	**3,15**	5,83	3,85	5,45	4,26	4,25
D30a	1,61	2,55	1,51	1,56	2,55	**0,87**	2,48	2,59	2,37
D32a	2,68	5,61	2,63	2,94	2,88	**1,97**	115	12,8	3,39
D32b	1,66	1,89	1,87	**1,56**	1,89	2,65	3,52	1,69	2,71
D80a	1,34	0,79	1,22	1,31	1,52	1,71	1,31	1,98	**0,65**
D80b	1,80	1,75	2,08	1,63	1,41	**0,75**	0,91	**0,75**	0,97
D82a	2,08	1,35	1,54	1,71	1,52	1,15	**0,90**	1,46	1,56
D82b	1,01	1,02	0,91	**0,69**	1,69	1,59	1,48	0,94	0,97
Total	1,87%	2,06%	1,87%	**1,67%**	2,34%	1,89%	9,90%	2,67%	1,86%

N. Dupin, Tighter MIP formulations for the discretised unit commitment problem with min-stop ramping constraints, EURO Journal of Computational Optimization, 2017.

Dual bounds at the root node before branching

	Lv2	Lv2b	Lv3	Lv3b	St2	St2b	St3	St3b	Mx3
D6	0,24	0,32	0,22	0,27	0,15	0,15	0,14	0,19	**0,12**
D13	1,60	1,48	1,67	1,59	1,37	1,48	**1,19**	1,23	1,38
D16	1,78	2,38	1,57	1,9	0,46	0,47	0,61	0,46	**0,25**
D35	0,92	0,61	0,62	0,68	0,70	0,78	0,65	0,79	**0,39**
D48	2,53	1,65	2,23	2,37	1,08	1,14	0,98	**0,87**	1,78
D67	0,77	0,64	0,77	0,67	0,30	0,24	0,20	**0,19**	0,62
D30a	0,63	0,59	0,63	0,62	0,39	0,47	**0,34**	**0,34**	0,38
D32a	0,65	**0,57**	0,59	0,62	0,69	0,65	0,68	0,68	0,59
D32b	0,88	**0,79**	0,86	0,92	0,93	0,92	0,91	0,88	0,89
D80a	1,34	1,09	1,34	1,30	0,67	0,69	**0,35**	0,50	1,23
D80b	0,31	0,33	0,28	0,30	0,12	0,09	0,10	**0,05**	0,23
D82a	0,61	0,58	0,60	0,60	0,57	0,56	0,53	**0,50**	0,60
D82b	0,64	0,62	0,64	0,64	0,35	0,34	0,33	**0,29**	0,62
Total	0,88%	0,76%	0,85%	0,87%	0,57%	0,58%	0,49%	**0,48%**	0,75%

N. Dupin, Tighter MIP formulations for the discretised unit commitment problem with min-stop ramping constraints, EURO Journal of Computational Optimization, 2017.

Quelques mots sur la parallélisation

- Initialisation à la racine (preprocessing, calcul PL, coupes) plutôt séquentielle
- Avec les branchements, algorithme B&B fortement parallélisable: travail sur les noeuds indépendants
- Bcp de solveurs (comme Cplex) ont par défaut une implémentation parallèle à mémoire partagée.
- Implémentation distribuée (type MPI) adaptée: permet de traiter un grand nombre de noeuds en même temps, peu de données à communiquer des données vs temps plus important de traitement par noeuds.
- Parallélisation GPGPU? Uniquement sur des cas très particuliers

`https://www.ibm.com/support/knowledgecenter/SSSA5P_12.6.3/ilog.odms.cplex.help/CPLEX/UsrMan/topics/parallel_optim/distribMIP/02_intro.html`

Autres types de relaxations

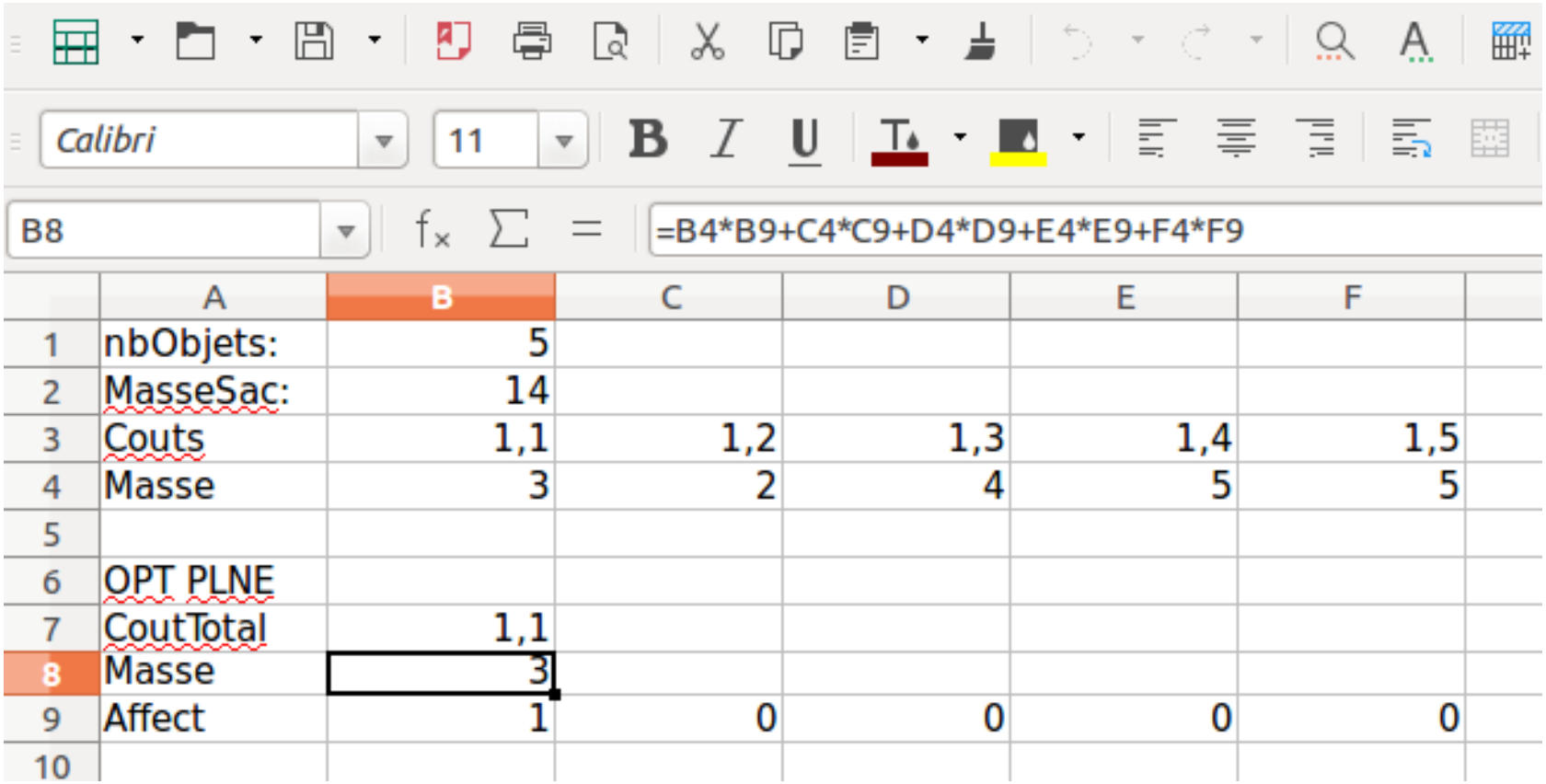
- Le principe de l'algorithme B&B peut être utilisé dès qu'une borne duale peut être calculée sur une structure arborescente.
- La relaxation continue et le calcul PL ont été utilisés jusqu'ici "LP-based Branch&Bound".
- D'autres bornes peuvent être calculées, et amener une résolution par l'algorithme de séparation-évaluation:
  - Bornes "combinatoires" spécifiques: au prochain cours
  - Bornes obtenues pour des PL avec un nombre exponentiel de contraintes (comme la formulation du TSP avec les sous-tours): au prochain cours
  - Bornes de relaxation Lagrangienne (CM12)
  - Reformulation de Dantzig-Wolfe et bornes de génération de colonnes, approche Branch-and-Price (CM12)

Quelques mots sur les extensions non-linéaires

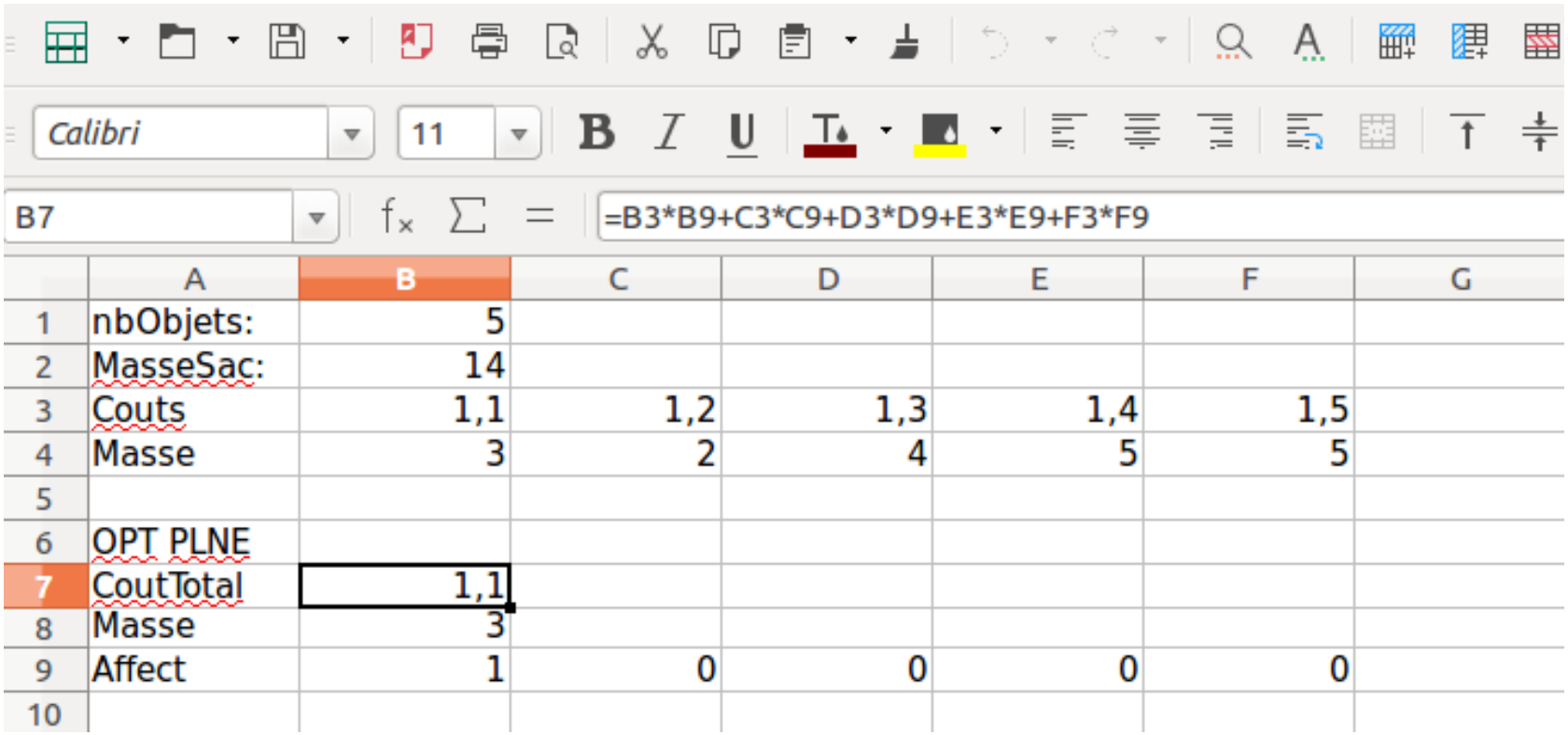
- Principes de l'algorithme B&B sont étendus dans cadre non linéaire en nombre entiers.
- Extensions naturelles d'heuristiques utilisées dans le cadre linéaire à l'intérieur du B& B.
- Avec des fonctions convexes ou concaves, MILP peut donner des bornes et schémas d'approximation.
- Difficulté dans calculs de bornes, induit des cas très spécifiques:
  - MIQP: fonction objectif quadratique, contraintes linéaires.
  - MIQCP: Mixed Integer Quadratically Constrained Program.
  - MINLP: Mixed Integer Non Linear Program.
  - SOCP: Second Order Cone Program.
  - SDP: Semi Definite Program.

Plan

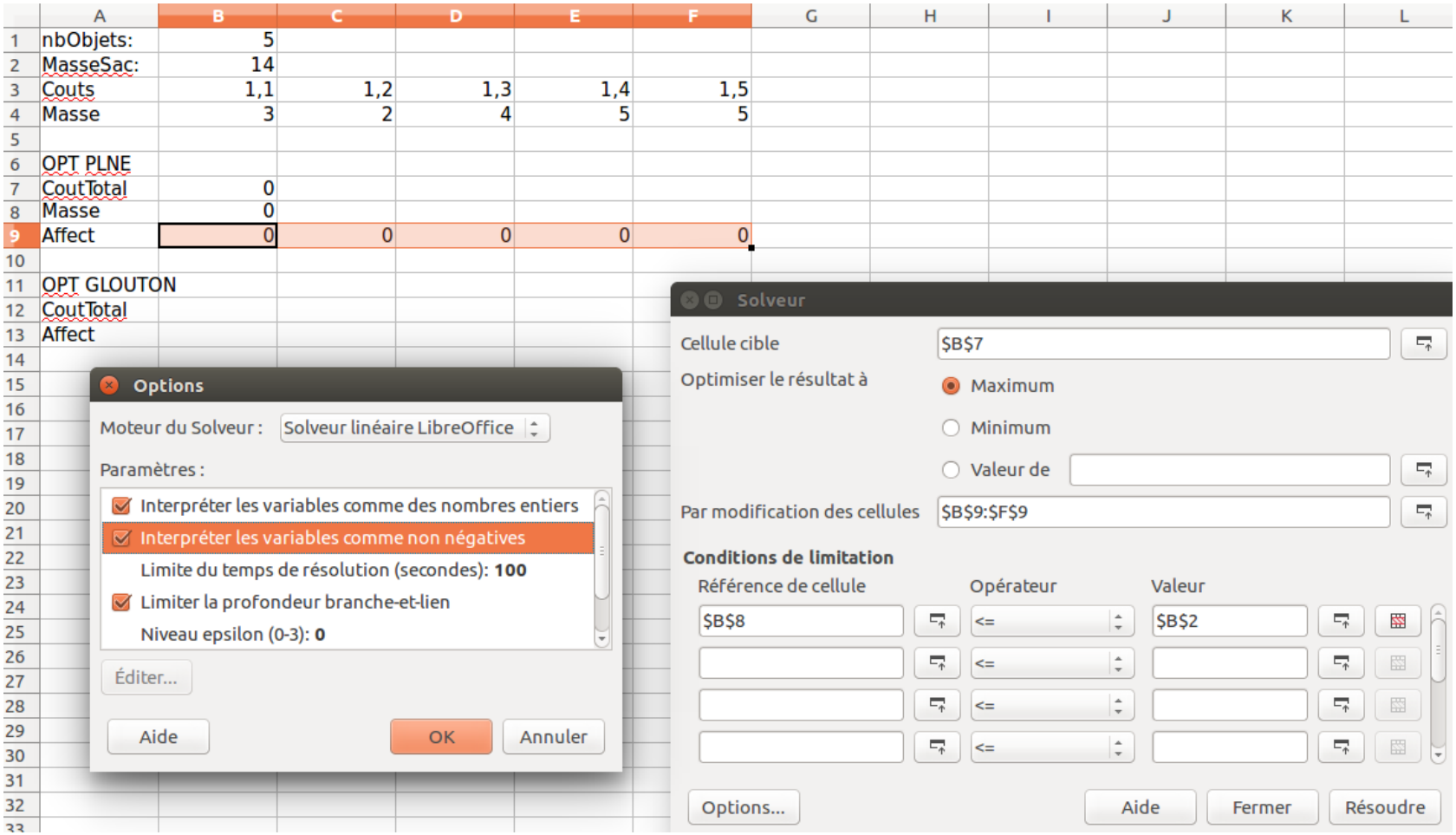
DEMO Excel, sur problèmes de sac à dos et d'affectation

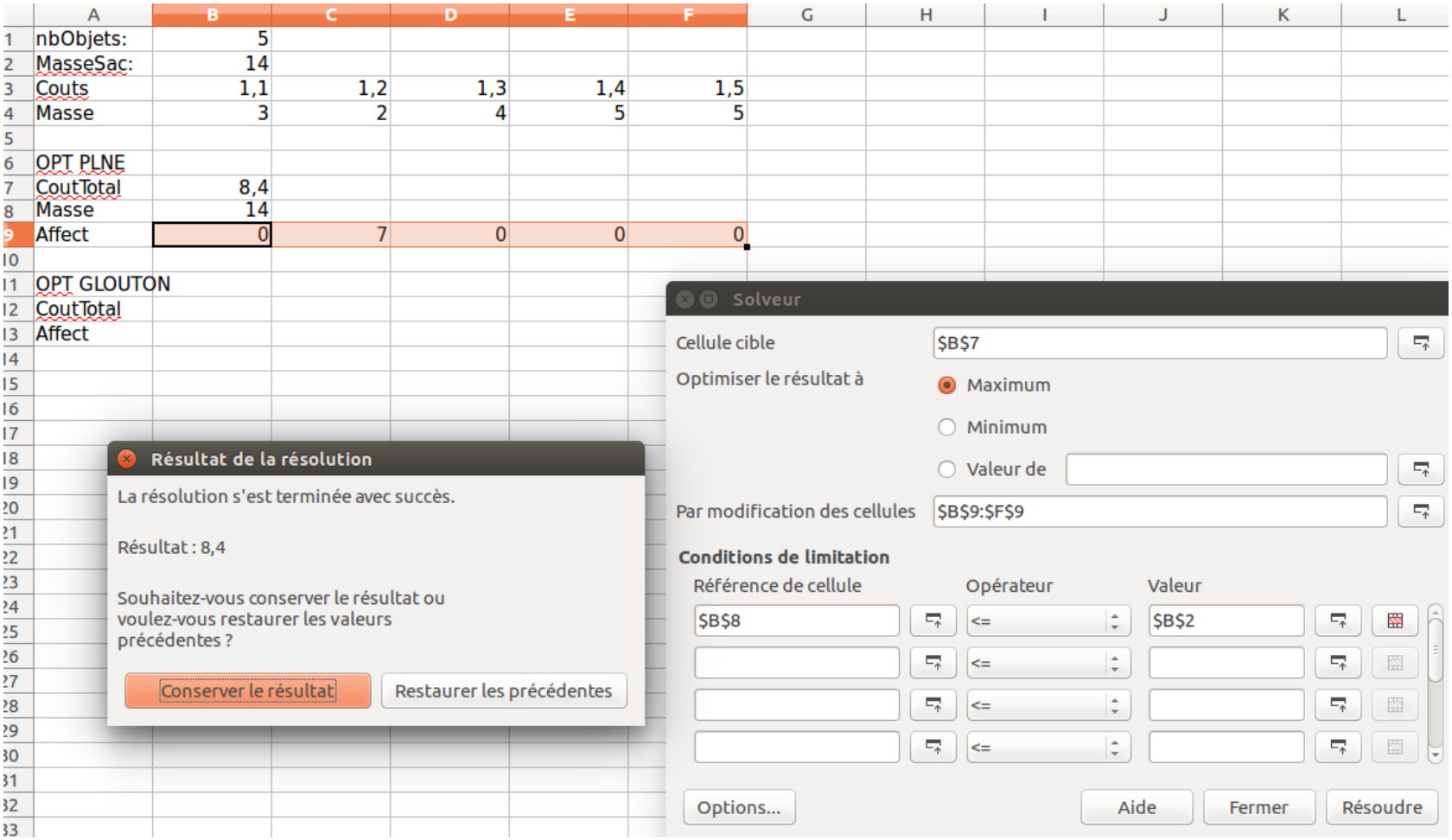


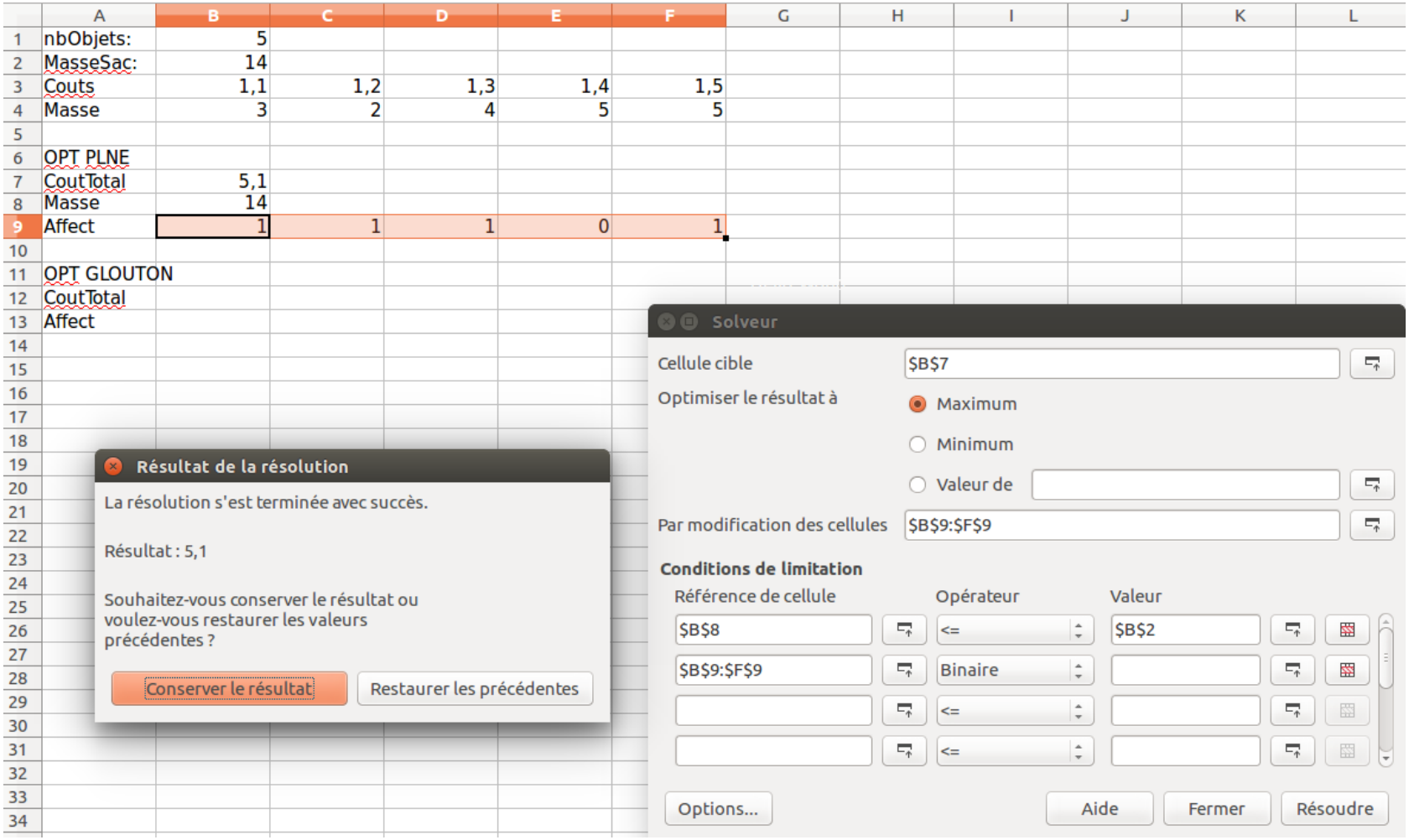
	A	B	C	D	E	F
1	nbObjets:	5				
2	MasseSac:	14				
3	Couts	1,1	1,2	1,3	1,4	1,5
4	Masse	3	2	4	5	5
5						
6	OPT PLNE					
7	CoutTotal	1,1				
8	Masse	3				
9	Affect	1	0	0	0	0
10						



	A	B	C	D	E	F	G
1	nbObjets:	5					
2	MasseSac:	14					
3	Couts	1,1	1,2	1,3	1,4	1,5	
4	Masse	3	2	4	5	5	
5							
6	OPT PLNE						
7	CoutTotal	1,1					
8	Masse	3					
9	Affect	1	0	0	0	0	
10							


nbObjets: 5
MasseSac: 14
Couts 1,1 1,2 1,3 1,4 1,5
Masse 3 2 4 5 5
OPT PLNE
CoutTotal 0
Masse 0
Affect 0 0 0 0 0
OPT GLOUTON
CoutTotal
Affect
Options
Moteur du Solveur : Solveur linéaire LibreOffice
Paramètres :
Interpréter les variables comme des nombres entiers
Interpréter les variables comme non négatives
Limite du temps de résolution (secondes): 100
Limiter la profondeur branche-et-lien
Niveau epsilon (0-3): 0
Éditer...
Aide
OK
Annuler
Solveur
Cellule cible $B$7
Optimiser le résultat à
Maximum
Minimum
Valeur de
Par modification des cellules $B$9:$F$9
Conditions de limitation
Référence de cellule
Opérateur
Valeur
$B$8 <= $B$2
Options...
Aide
Fermer
Résoudre


nbObjets: 5
MasseSac: 14
Couts 1,1 1,2 1,3 1,4 1,5
Masse 3 2 4 5 5
OPT PLNE
CoutTotal 8,4
Masse 14
Affect 0 7 0 0 0
OPT GLOUTON
CoutTotal
Affect
Résultat de la résolution
La résolution s'est terminée avec succès.
Résultat : 8,4
Souhaitez-vous conserver le résultat ou voulez-vous restaurer les valeurs précédentes ?
Conserver le résultat
Restaurer les précédentes
Solveur
Cellule cible
$B$7
Optimiser le résultat à
Maximum
Minimum
Valeur de
Par modification des cellules
$B$9:$F$9
Conditions de limitation
Référence de cellule
Opérateur
Valeur
$B$8
<=
$B$2
Options...
Aide
Fermer
Résoudre


nbObjets: 5
MasseSac: 14
Couts 1,1 1,2 1,3 1,4 1,5
Masse 3 2 4 5 5
OPT PLNE
CoutTotal 5,1
Masse 14
Affect 1 1 1 0 1
OPT GLOUTON
CoutTotal
Affect
Résultat de la résolution
La résolution s'est terminée avec succès.
Résultat : 5,1
Souhaitez-vous conserver le résultat ou voulez-vous restaurer les valeurs précédentes ?
Conserver le résultat
Restaurer les précédentes
Solveur
Cellule cible
$B$7
Optimiser le résultat à
Maximum
Minimum
Valeur de
Par modification des cellules
$B$9:$F$9
Conditions de limitation
Référence de cellule
Opérateur
Valeur
$B$8
<=
$B$2
$B$9:$F$9
Binaire
Options...
Aide
Fermer
Résoudre

Limites d'utilisation de tableurs Excel ou Open Office

- Excel: nombres de variables limitées sauf additif payant.
- Solveurs d'optimisation moins efficaces que solveurs commerciaux spécialisés en optimisation linéaire.
- Juste le résultat, pas de log, difficile de déboguer. Contraintes de temps: la solution fournie est elle optimale ?
- Modélisation avec peu de contraintes, ajoutées une à une sans VBA. Pas de langage formel au plus près des équations mathématiques. (Comment écrire maxClique/maxStable avec un grand graphe dans Excel?)
- Calcul unique, parfois besoin de faire successivement plusieurs calculs d'optimisation
- Comment faire avec des variables à 3 indices?

$\Longrightarrow$ Ok pour une utilisation très simple, limité pour de l'optimisation de problèmes plus complexes.

Solveurs de PL/PLNE et interfaces

- Commerciaux avec licence académique: Cplex, Xpress, Gurobi, SCIP.
- Libres: GLPK, suite COIN-OR (CBC pour PLNE, CLP pour PL).
- Les solveurs peuvent être appelés en API des langages de programmation les plus courants (C++, Java, Python, ...)
- "Modeleurs" avec interface vers les solveurs, pour une résolution "model&run" au plus proche de la modélisation mathématique. OPL pour Cplex, AMPL, GNU MathProg, ZIMPL ...
- Julia JuMP: bibliothèque Julia pour appeler différents solveurs + quelques fonctionnalités avancées (warmstart, callbacks) avec une modélisation générique. Extensions multi-objectifs et décompositions (génération de colonnes)
- PULP, python-MIP, Pyomo: API Python génériques pour appeler différents solveurs. Pyomo gère des solveurs non-linéaires, Python-MIP ne gère que CBC et Gurobi, mais a des fonctionalités avancées (warmstart, callbacks), et plus efficace que PuLP:

  `https://python-mip.readthedocs.io/en/latest/bench.html`

Fichier génériques .lp et .mps

- Des formats génériques de fichiers permettent de lire des PL/PLNE depuis n'importe quel solveur .lp et .mps
- Fichiers .mps peu intuitifs, contrairement aux .lp (qui sont un outil de débogage).
- Intérêt 1: Benchmark de solveurs sur des problèmes générés aux formats .lp, entre solveurs commerciaux, ou comparaison de performances solveur libre vs commercial
- Intérêt 2: les .lp et .mps peuvent être générés automatiquement. Les .lp peuvent être générés par des scripts très simples pour des problèmes comme MaxClique/MaxStable
- Intérêt 3: les .lp automatiquement générés peuvent servir pour déboguer sur une mauvaise écriture de PL/PLNE.

On trouvera des exemples de .lp et .mps dans le dossier:

/opt/ibm/ILOG/CPLEX_Studio1271/cplex/examples/data

Vue d'un fichier .lp

```
Maximize + 10a_1 + 10a_2 + 15a_3 + 20a_4 + 20a_5 + 24a_6 + 24a_7 + 50a_8

Subject To
 + 10a_1 + 10a_2 + 15a_3 + 20a_4 + 20a_5 + 24a_6 + 24a_7 + 50a_8 <= 100

Bounds
 0 <=a_1 <= 1
 0 <=a_2 <= 1
 0 <=a_3 <= 1
 0 <=a_4 <= 1
 0 <=a_5 <= 1
 0 <=a_6 <= 1
 0 <=a_7 <= 1
 0 <=a_8 <= 1

Generals
a_1
a_2
a_3
a_4
a_5
a_6
a_7
a_8

End
```

Appeler un solveur sur un fichier .lp

Ainsi on peut résoudre un problème d'optimisation en lisant un fichier .lp ou .mps par:

> glpsol –lp location.lp
> glpsol –mps afiro.mps

On peut même exporter un modèle dans un format .lp ou .mps par:

> glpsol –mps afiro.mps –wlp afiro.lp

Avec cbc (COIN-OR), la commande est simplement:

> cbc location.lp

Pour lancer l'optimisation d'un .lp avec Cplex, il faut préciser les commandes internes. On peut charger le modèle et résoudre l'optimisation et afficher les solutions en une commande par:

> cplex -c "read location.lp" "optimize" "display solution variables -"

Les modeleurs: séparer modèle et données

- Idée: séparer modèle et données
- Le modèle (.mod) définit les structures de données à récupérer dans le fichier de donnée, et écrit le modèle mathématique de manière générique suivant le format de données défini (.dat)
- Le .mod reproduit le modèle mathématique, dnas un langage au plus près des équations PLNE.
- Les .dat peuvent avoir une taille variable, indexée par des paramètres fournis au début du .dat. (ex: nombre d'objets dans un sac à dos, ou nombre de sommets et d'arêtes dans un graphe)
- Modeleurs: GNU MathProg (GLPK), ZIMPL, OPL et AMPL (langage propriétaire)

Un cours d'introduction au formalisme des .mod et .dat du modeleur GLPK: `http://lim.univ-reunion.fr/staff/fred/Enseignement/Optim/doc/GLPK/CoursGLPK.pdf`

Illustration séparation .mod et .dat GLPK

Ouvrir knapsack.mod ~/Bureau/Enseignement... Enregistrer

```
# This file shows how to model a knapsack problem in GMPL.

# Size of knapsack
param c;

# Items: index, size, profit
set I, dimen 3;

# Indices
set J := setof{(i,s,p) in I} i;

# Assignment
var a{J}, binary;

maximize obj :
  sum{(i,s,p) in I} p*a[i];

s.t. size :
  sum{(i,s,p) in I} s*a[i] <= c;

solve;

printf "The knapsack contains:\n";
printf {(i,s,p) in I: a[i] == 1} " %i", i;
printf "\n";

end;
```

Paramètres

Déclaration des variables

Fonction objectif

Ecriture des contraintes PL/PLNE
utilise les variables définies dans .mod

Ouvrir knapsack.dat ~/Bureau/Enseignem...

```
data;

# Size of the knapsack
param c := 100;

# Items: index, size, profit
set I :=
  1 10 10
  2 10 10
  3 15 15
  4 20 20
  5 20 20
  6 24 24
  7 24 24
  8 50 50;

end;
```

Paramètres de résolution PLNE avec GLPK

```
(base) nd@nd-Latitude-E6430:~/Bureau/Enseignement/Optim-RO/OptimComb-M1/TP-PLNE/TP2/GLPK$ glpsol -h
Usage: glpsol [options...] filename

General options:
   --mps             read LP/MIP problem in fixed MPS format
   --freemps         read LP/MIP problem in free MPS format (default)
   --lp              read LP/MIP problem in CPLEX LP format
   --glp             read LP/MIP problem in GLPK format
   --math            read LP/MIP model written in GNU MathProg modeling
                     language
   -m filename, --model filename
                     read model section and optional data section from
                     filename (same as --math)
   -d filename, --data filename
                     read data section from filename (for --math only);
                     if model file also has data section, it is ignored
   -y filename, --display filename
                     send display output to filename (for --math only);
                     by default the output is sent to terminal
   --seed value      initialize pseudo-random number generator used in
                     MathProg model with specified seed (any integer);
                     if seed value is ?, some random seed will be used
   --mincost         read min-cost flow problem in DIMACS format
   --maxflow         read maximum flow problem in DIMACS format
   --cnf             read CNF-SAT problem in DIMACS format
   --simplex         use simplex method (default)
   --interior        use interior point method (LP only)
   -r filename, --read filename
                     read solution from filename rather to find it with
                     the solver
   --min             minimization
   --max             maximization
   --scale           scale problem (default)
   --noscale         do not scale problem
   -o filename, --output filename
                     write solution to filename in printable format
   -w filename, --write filename
                     write solution to filename in plain text format
   --ranges filename
                     write sensitivity analysis report to filename in
                     printable format (simplex only)
   --tmlim nnn       limit solution time to nnn seconds
   --memlim nnn      limit available memory to nnn megabytes
   --check           do not solve problem, check input data only
   --name probname   change problem name to probname
   --hide            remove all symbolic names from problem object
   --wmps filename   write problem to filename in fixed MPS format
   --wfreemps filename
                     write problem to filename in free MPS format
   --wlp filename    write problem to filename in CPLEX LP format
   --wglp filename   write problem to filename in GLPK format
   --wcnf filename   write problem to filename in DIMACS CNF-SAT format
   --log filename    write copy of terminal output to filename
   -h, --help        display this help information and exit
```

```
Options specific to simplex solver:
   --primal          use primal simplex (default)
   --dual            use dual simplex
   --std             use standard initial basis of all slacks
   --adv             use advanced initial basis (default)
   --bib             use Bixby's initial basis
   --ini filename    use as initial basis previously saved with -w
                     (disables LP presolver)
   --steep           use steepest edge technique (default)
   --nosteep         use standard "textbook" pricing
   --relax           use Harris' two-pass ratio test (default)
   --norelax         use standard "textbook" ratio test
   --flip            use long-step ratio test
   --presol          use presolver (default; assumes --scale and --adv)
   --nopresol        do not use presolver
   --exact           use simplex method based on exact arithmetic
   --xcheck          check final basis using exact arithmetic

Options specific to interior-point solver:
   --nord            use natural (original) ordering
   --qmd             use quotient minimum degree ordering
   --amd             use approximate minimum degree ordering (default)
   --symamd          use approximate minimum degree ordering

Options specific to MIP solver:
   --nomip           consider all integer variables as continuous
                     (allows solving MIP as pure LP)
   --first           branch on first integer variable
   --last            branch on last integer variable
   --mostf           branch on most fractional variable
   --drtom           branch using heuristic by Driebeck and Tomlin
                     (default)
   --pcost           branch using hybrid pseudocost heuristic (may be
                     useful for hard instances)
   --dfs             backtrack using depth first search
   --bfs             backtrack using breadth first search
   --bestp           backtrack using the best projection heuristic
   --bestb           backtrack using node with best local bound
                     (default)
   --intopt          use MIP presolver (default)
   --nointopt        do not use MIP presolver
   --binarize        replace general integer variables by binary ones
                     (assumes --intopt)
   --fpump           apply feasibility pump heuristic
   --proxy [nnn]     apply proximity search heuristic (nnn is time limit
                     in seconds; default is 60)
   --gomory          generate Gomory's mixed integer cuts
   --mir             generate MIR (mixed integer rounding) cuts
   --cover           generate mixed cover cuts
   --clique          generate clique cuts
   --cuts            generate all cuts above
   --mipgap tol      set relative mip gap tolerance to tol
   --minisat         translate integer feasibility problem to CNF-SAT
                     and solve it with MiniSat solver
```

Paramètres de résolution PLNE avec GLPK, exemples

Résoudre un .mod et un .dat:

```
> glpsol -m knapsack.mod -d knapsack.dat
```

Juste pour générer le fichier .lp sans résoudre:

```
> glpsol -m knapsack.mod -d knapsack.dat --check --wlp knapsack.lp
```

Résoudre knapsack.lp sur 1h de temps limite:

```
> glpsol --lp knapsack.lp --tmlim 3600
```

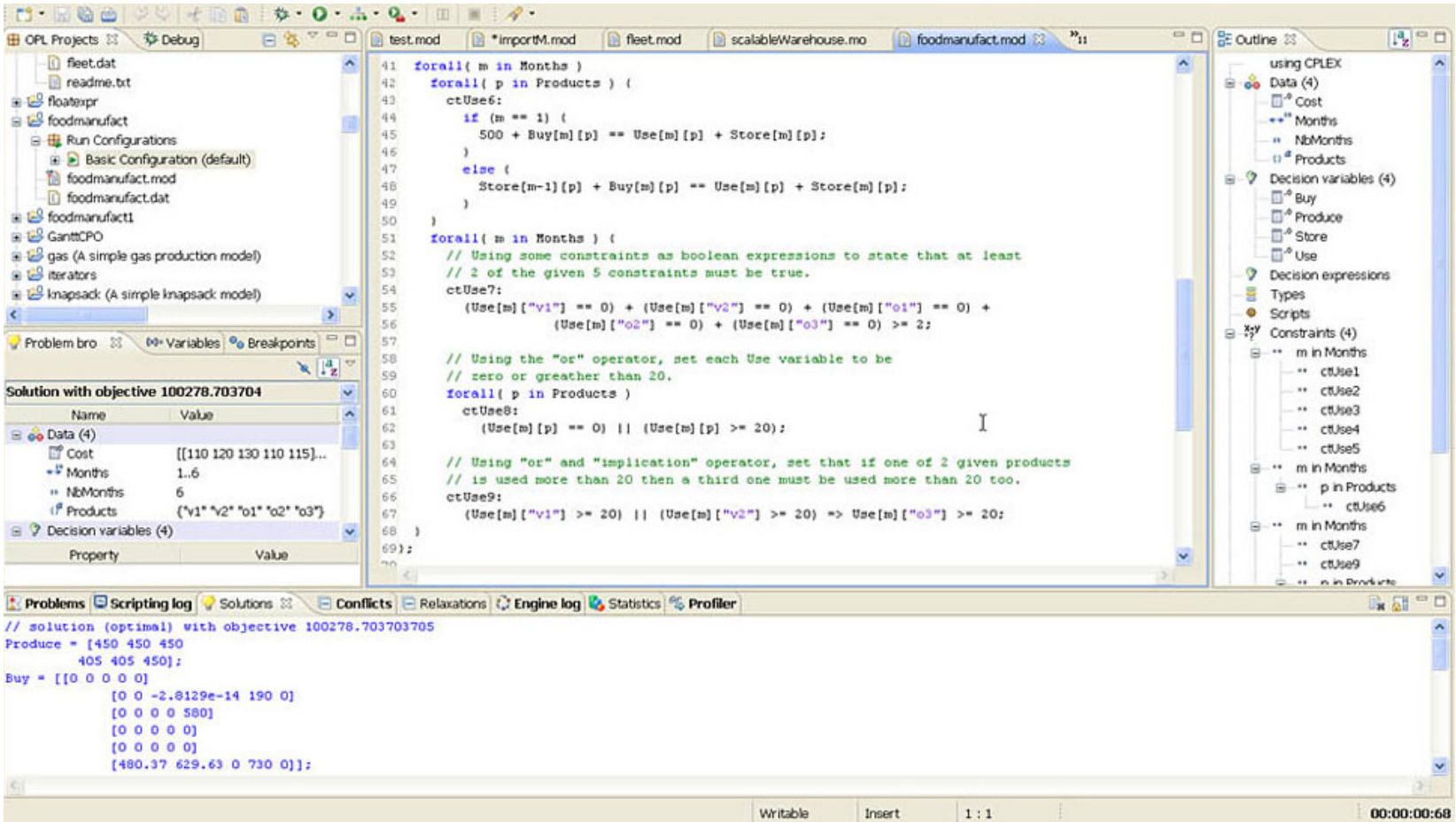
Résoudre knapsack.lp en activant les coupes de couvertures:

```
> glpsol --lp knapsack.lp --cover
```

Résoudre knapsack.lp et afficher le résultat sur un fichier solution.txt:

```
> glpsol --lp knapsack.lp -o solution.txt
```

Vue de OPL IDE



Appeler OPL

L'IDE d'OPL (idéal pour déboguer) s'appelle par:

```
> oplide
```

OPL s'appelle en ligne de commande avec un modèle .mod et un ou plusieurs fichiers de données .dat par:

```
> oplrun modelOPL.mod   data1.dat (facult: data2.dat data3.dat)
```

Intérêt: ne charge pas l'interface graphique et l'environnement de débogage.

ATTENTION: pas de format commun .mod ou .dat entre GLPK et OPL

Imports de données dans OPL

On trouvera des exemples de projets OPL (.mod et .dat) dans le dossier:

/opt/ibm/ILOG/CPLEX_Studio1271/opl/examples/opl

L'exemple oil correspond au mélange de carburants du TD1

L'exemple oil illustre comment:

- un .dat peut être généré classiquement, suivant une structure de données définies dans un .mod (oil.mod et oil.dat)
- un .dat peut pointer sur des données Excel (oil.mod, oilSheet.dat et oilSheet.xls)
- suivant les versions de Cplex et OPL, un import depuis une base de données Access.

N.B: GLPK a aussi un mode d'import de données d'un tableur en .csv

Paramétrer la résolution Cplex depuis OPL

OPL a un langage de script qui permet d'effectuer des pré-traitement et post traitements.

On peut modifier les paramètres de résolution Cplex en en-tête d'un .mod OPL

execute SET_PARAM_CPLEX {
cplex.tilim = 300; temps de résolution max
cplex.nodelim = 1000; // nombre de noeuds BandB max
}

Pour la liste des paramètres de résolution Cplex:

`https://www.ibm.com/support/knowledgecenter/SSSA5P_12.7.1/ilog.odms.cplex.help/CPLEX/Parameters/topics/introListAlpha.html`

Savoir lire les logs de Cplex (1)

```
CPXPARAM_TimeLimit                               3600
Tried aggregator 3 times.
MIP Presolve eliminated 160065 rows and 102404 columns.
MIP Presolve modified 234 coefficients.
Aggregator did 95 substitutions.
Reduced MIP has 7072 rows, 18202 columns, and 121782 nonzeros.
Reduced MIP has 1904 binaries, 0 generals, 0 SOSs, and 0 indicators.
Presolve time = 0,71 sec. (514,12 ticks)
Found incumbent of value 1,7055733e+13 after 1,11 sec. (833,51 ticks)
Probing fixed 69 vars, tightened 71 bounds.
Probing changed sense of 80 constraints.
Probing time = 0,08 sec. (25,23 ticks)
Cover probing fixed 0 vars, tightened 12 bounds.
Tried aggregator 2 times.
MIP Presolve eliminated 352 rows and 199 columns.
MIP Presolve modified 15 coefficients.
Aggregator did 8 substitutions.
Reduced MIP has 6712 rows, 17995 columns, and 118857 nonzeros.
Reduced MIP has 1765 binaries, 0 generals, 0 SOSs, and 0 indicators.
Presolve time = 0,13 sec. (98,80 ticks)
Probing time = 0,05 sec. (17,03 ticks)
Cover probing fixed 0 vars, tightened 5 bounds.
Clique table members: 40614.
Tightened 2 constraints.
MIP emphasis: balance optimality and feasibility.
MIP search method: dynamic search.
Parallel mode: deterministic, using up to 4 threads.
Root relaxation solution time = 1,55 sec. (1049,35 ticks)

        Nodes                                         Cuts/
   Node  Left     Objective  IInf  Best Integer    Best Bound    ItCnt     Gap

*     0+    0                       1,70557e+13   1,72693e+10             99,90%
      0     0   1,01431e+11   831   1,70557e+13   1,01431e+11       53   99,41%
      0     0   1,01864e+11   795   1,70557e+13     Cuts: 239      669   99,40%
*     0+    0                       1,02709e+11   1,01864e+11             0,82%
*     0+    0                       1,02475e+11   1,01864e+11             0,60%
      0     0   1,02008e+11   716   1,02475e+11     Cuts: 239     1138    0,46%
*     0+    0                       1,02469e+11   1,02008e+11             0,45%
      0     0   1,02040e+11   774   1,02469e+11     Cuts: 196     1383    0,42%
      0     0   1,02046e+11   787   1,02469e+11     Cuts: 138     1548    0,41%
      0     0   1,02110e+11   787   1,02469e+11      Cuts: 83     1821    0,35%
      0     0   1,02119e+11   760   1,02469e+11     Cuts: 102     1973    0,34%
      0     0   1,02120e+11   850   1,02469e+11      Cuts: 58     2045    0,34%
      0     0   1,02120e+11   778   1,02469e+11      Cuts: 65     2102    0,34%
      0     0   1,02120e+11   800   1,02469e+11      Cuts: 25     2130    0,34%
*     0+    0                       1,02468e+11   1,02120e+11             0,34%
      0     2   1,02120e+11   800   1,02468e+11   1,02120e+11     2130    0,34%
Elapsed time = 11,97 sec. (8496,47 ticks, tree = 0,02 MB, solutions = 3)
      2     4   1,02181e+11   688   1,02468e+11   1,02121e+11     2842    0,34%
      9     9   1,02246e+11   702   1,02468e+11   1,02141e+11     3844    0,32%
     16    11   1,02378e+11   666   1,02468e+11   1,02141e+11     4750    0,32%
```

Paramètre de temps de résolution activé: 3600 s

Une solution réalisable est trouvée (heuristique primale): qualité moyenne a posteriori

Le preprocessing de Cplex réduit le problème (preprocessing, élimination variables inutiles ...):

Calcul de relaxation continue: permet d'avoir une première borne inférieure à 1,01431.10^11

phase ou Cplex améliore les bornes duales avec des coupes sans branchement, et également des bornes primales avec des heuristiques.

Un gap de 0,34 % entre borne sup et inf avant tout branchement

La phase de branchement commence ...

Savoir lire les logs de Cplex (2)

```
Elapsed time = 107,55 sec. (50893,66 ticks, tree = 1,23 MB, solutions = 21)
   2696   114   1,02379e+11   676   1,02426e+11   1,02395e+11   152060    0,03%
   2753   127   1,02369e+11   568   1,02426e+11   1,02395e+11   153516    0,03%
   2852   173   1,02345e+11   539   1,02426e+11   1,02395e+11   162554    0,03%
*  2901+  240                       1,02425e+11   1,02395e+11              0,03%
*  2916+  250                       1,02423e+11   1,02395e+11              0,03%
*  2936+  250                       1,02422e+11   1,02395e+11              0,03%
   2940   191   1,02417e+11   323   1,02422e+11   1,02395e+11   167605    0,03%
   3077   279        cutoff         1,02422e+11   1,02395e+11   176896    0,03%
   3158   371   1,02404e+11   601   1,02422e+11   1,02395e+11   183639    0,03%
   3213   369   1,02421e+11   413   1,02422e+11   1,02395e+11   189863    0,03%
   3300   382        cutoff         1,02422e+11   1,02395e+11   192274    0,03%
   3366   412   1,02417e+11   298   1,02422e+11   1,02395e+11   198952    0,03%
   3428   429   1,02413e+11   436   1,02422e+11   1,02395e+11   209478    0,03%
Elapsed time = 132,30 sec. (60728,72 ticks, tree = 11,78 MB, solutions = 24)
   3507   427        cutoff         1,02422e+11   1,02395e+11   214160    0,03%
   3583   404        cutoff         1,02422e+11   1,02395e+11   219911    0,03%
   3652   388   1,02420e+11   603   1,02422e+11   1,02395e+11   225764    0,03%
   3753   360   1,02417e+11   467   1,02422e+11   1,02409e+11   231460    0,01%

Clique cuts applied:  1
Implied bound cuts applied:  29
Flow cuts applied:  31
Mixed integer rounding cuts applied:  217
Zero-half cuts applied:  16
Gomory fractional cuts applied:  12

Root node processing (before b&c):
  Real time             =   11,90 sec. (8443,63 ticks)
Parallel b&c, 4 threads:
  Real time             =  130,35 sec. (56469,87 ticks)
  Sync time (average)   =    8,10 sec.
  Wait time (average)   =    0,01 sec.
                          ------------
Total (root+branch&cut) =  142,25 sec. (64913,50 ticks)
OBJECTIF: 1.024223477e+11
```

En 108s, 2696 noeuds explorés,
(ou en cours d'exploration)
gap de 0.03 % entre borne primale et borne duale

L'algorithme de Branch & Bound a convergé
avec une tolérance par défaut de 0,01%
l'optimum est prouvé à 0,01 % près

Types et volumes de coupes générés par Cplex
(info utile pour comparer plusieurs modélisations
PLNE d'un même problème)

Modeleurs vs limites d'utilisation de tableurs

- Pas de limitation du nombres de variables (algo exponentiel délimite ce qui est praticable ou non).
- Peut être utilisé pour définir des .lp, qui permettront d'utiliser des solveurs libres et/ou commerciaux spécialisés.
- Sorties .log montrent les caractéristiques de convergence B&B, permet des analyses de performances de différentes étapes clés de B&B.
- Export de .lp aide à déboguer.
- Modélisation avec du langage formel au plus près des équations mathématiques. (facile d'écrire maxClique/maxStable)
- Peut être appelé en console ou en API pour des calculs itérés.
- Variables à 1,2, 3 ou plus d' indices: aucune difficulté de modélisation avec langage formel

Plan

On a vu sur ce cours du point de vue pratique

- Résolution PL: utiliser les solveurs disponibles.
- Résolution PLNE: algorithme Branch&Bound de recherche arborescente, complexité exponentielle en général.
- Branch&Bound: accélération empirique avec de bonnes stratégies de coupes et d'heuristiques primales. La connaissance du problème spécifique ou une meilleure modélisation PLNE permettent d'accélérer la convergence
- Plusieurs solveurs libres et commerciaux présentés, plusieurs cadres d'utilisation (API, modeleur, depuis un tableur)

Ce cours a répondu à plusieurs questions en suspens

- Résolution générique dans les solveurs PLNE, et des éléments pour l'efficacité pratique du solveur.
- L'utilisation des coupes de cliques dans un solveur générique de PLNE.
- L'impact d'avoir une formulation PLNE fournissant une meilleure relaxation continue.
- Pourquoi il est important de limiter les symétries dans la résolution PLNE
- Plusieurs solveurs libres et commerciaux présentés, plusieurs cadres d'utilisation (API, modeleur, depuis un tableur)

Pour la suite

Prochains TP:

- Expérimenter les variantes d'algorithmes avec le code C++ spécifique au sac à dos
- TP GLPK, pour expérimenter l'utilisation d'un solveur générique

Prochains cours:

- Quelles variantes d'algorithmes de recherche arborescente pour l'optimisation combinatoire?
- En particulier, comment résoudre un PLNE avec un nb non énumérable de contraintes?
- Comment résoudre un PLNE avec un nb non énumérable de variables? (ex: coloration)
- Recherche arborescente en IA, des cousins de l'algorithme de Branch&Bound ?