RECHERCHE OPÉRATIONNELLE AVAINCÉE 2

Enseignant:
M. MAABED Mohammed

Chapitre III

Programmation linéaire en nombres entiers

mai 2026



Plan du chapitre

01

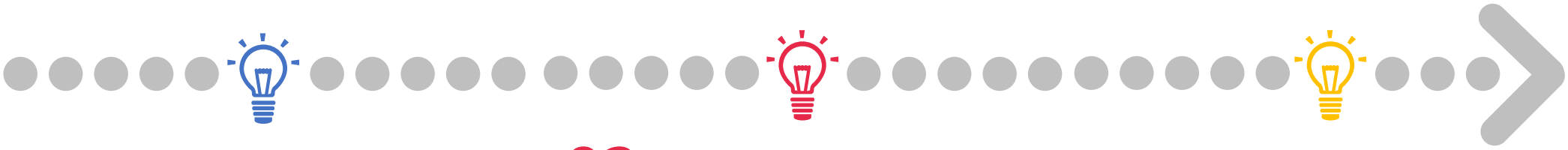
Méthode Branch & Bound

Une technique d'optimisation qui explore l'espace des solutions en divisant le problème en sous-problèmes (**branch**) et en éliminant ceux qui ne peuvent pas mener à une meilleure solution (**bound**). Elle permet ainsi de réduire les recherches tout en garantissant la solution optimale.

03

Méthode Branch & Cut

Combine le principe du Branch and Bound avec l'ajout de coupes (plans sécants). Elle explore l'espace des solutions en créant des sous-problèmes tout en ajoutant des contraintes supplémentaires pour éliminer les solutions fractionnaires. Cette combinaison réduit efficacement l'espace de recherche et permet d'atteindre plus rapidement une solution entière optimale.



02

Méthode de plans scécants (coupe de gomory)

Elle consiste à ajouter au modèle des plans (coupes) qui éliminent les solutions fractionnaires tout en conservant toutes les solutions entières possibles. À chaque itération, ces nouveaux plans resserrent progressivement la région réalisable jusqu'à obtenir une solution entière optimale.

Contexte Général

Dans de nombreux problèmes réels, **la programmation linéaire classique** permet de trouver des solutions optimales, mais celles-ci ne sont pas toujours exploitables **dans la pratique**. En effet, les modèles linéaires supposent que les variables de décision peuvent prendre n'importe quelle valeur réelle, y compris des fractions. Or, dans le monde réel, beaucoup de décisions **ne peuvent pas** être fractionnées.



- On ne peut pas affecter **3,5 employés** à une équipe.
- On ne peut pas réserver **2,7 serveurs** pour un service en ligne.
- On ne peut pas activer **0,4 connexion réseau**.
- On ne peut pas ouvrir **0,4 usine**.

Contexte Général



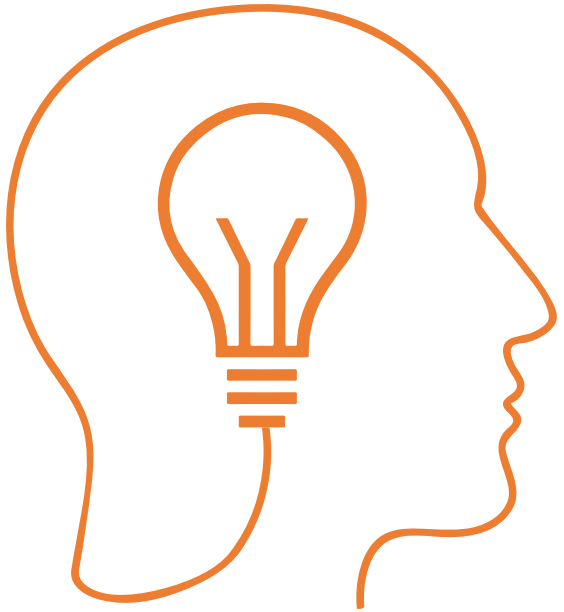
1 | Il peut sembler que les PLNE sont plus faciles à résoudre car ils ont peu de solutions à considérer.

2 | Même avec un nombre fini de solutions, le total peut être extrêmement grand. Par exemple, dans un PLNE avec $\boldsymbol{n}$ variables binaires, il existe 2^n solutions possibles à considérer. Ainsi, pour $n = 20$, on obtient plus d'un million de solutions.

3 | Le fait d'éliminer certaines solutions faisables (non entières) ne rend pas la résolution plus facile. Au contraire, c'est précisément grâce à ces solutions que l'on peut garantir l'existence d'une solution optimale.

4 | La plupart des algorithmes de résolution des ILP intègrent la méthode du simplexe en ignorant, dans un premier temps, les contraintes d'intégralité. Cette approche est appelée **relaxation**.

Contexte Général



On pourrait penser que, pour avoir des valeurs entières à ce problème, il suffit de faire un arrondi de la solution réelle obtenue.

Cette technique d'arrondissement peut fournir directement la solution optimale pour certains cas.

Pour d'autres cas, cette combine par arrondissement serait loin d'être la solution optimale ou serait même très éloignée, voire désastreuse par rapport à la solution optimale apportée par la résolution d'un PLNE.

Exemple:

On considère le PL suivant :

$$(P)\begin{cases} Z_{max} = 2x_1 + 1.28x_2 \\ 100x_1 + 62x_2 \leq 100 \\ 6x_1 - 4x_2 \geq -8 \\ \\ x_1,\ x_2 \in \mathrm{N} \end{cases}$$

- Solution optimale du PL:
  x^*_{PL} = (1.95 , 4.92); Z_{PL} = 5.098
- Solution arrondie du PL:
  x^*_{AR} = (2 , 5); Z_{AR} = 10
- Solution optimale du PLNE:
  x^*_{NE} = (5 , 0); Z_{NE} = 5

Problèmes Classiques de la PLNE

Problème de sac à dos

On dispose d'un sac de capacité maximale C et de n objets.
Chaque objet i est caractérisé par :

- un **poids** w_i , qui représente l'encombrement ou la quantité de ressource qu'il consomme,
- une **valeur** v_i , qui mesure l'intérêt ou le gain que l'on obtient en choisissant cet objet.

L'objectif est de **sélectionner un sous-ensemble d'objets** de manière à : **maximiser l'utilité totale** transportée dans le sac, tout en respectant la **contrainte de capacité** : le poids total ne doit pas dépasser C.



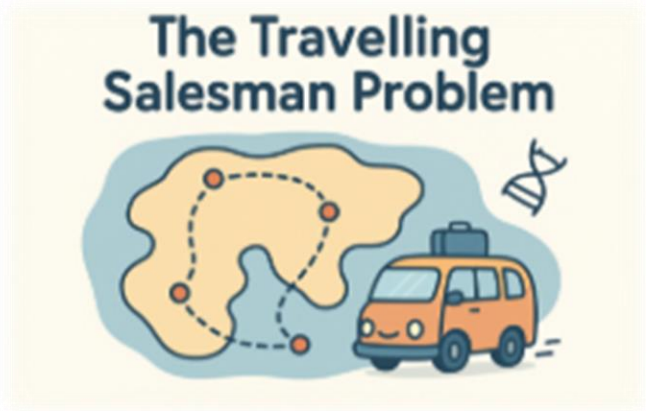
Formulation en PLNE

$$\begin{cases} \sum_{i=1}^{n} v_i x_i & \leftarrow \text{Maximisation de l'utilité totale} \\ \sum_{i=1}^{n} w_i x_i \leq W & \leftarrow \text{Limitation du poids totale} \\ x_i \in \{0,1\} \end{cases}$$

Problème du voyageur de commerce (Travelling Salesman Problem, TSP)

Un voyageur doit visiter n villes, en passant **une seule fois par chaque ville**, et revenir à sa ville de départ. On cherche le circuit de longueur minimale.

- c_{ij} = coût (ou distance) pour aller de la ville i à la ville j.
- x_{ij} = 1 si l'arc (i, j) est emprunté, 0 sinon.



Formulation en PLNE

$$\begin{cases} min \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} \\ \sum_{i=1}^{n} x_{ij} = 1, \forall i \quad \sum_{j=1}^{n} x_{ij} = 1, \forall j \\ \sum_{i\in S}\sum_{j\in S} x_{ij} \le |S| - 1, \forall S \subset \{1,..,n\}, 2 \le |S| \le n-1 \\ x_{ij} \in \{0,1\} \end{cases}$$

← min $(c_{11}x_{11} + c_{12}x_{12} + c_{13}x_{13} + \cdots + c_{1n}x_{1n} + c_{21}x_{21} + c_{22}x_{22} + \cdots + c_{2n}x_{2n} + \cdots + c_{n1}x_{n1} + \cdots + c_{nn}x_{nn})$

← Chaque ville est quittée 1 fois et visitée 1 fois

← Contrainte d'élimination de cycles

Contrainte d'élimination de cycles (Exemple):

Pour S = {1,2,3} , pour 3 sommets et pour avoir un cycle il faut compter **3 arcs**.

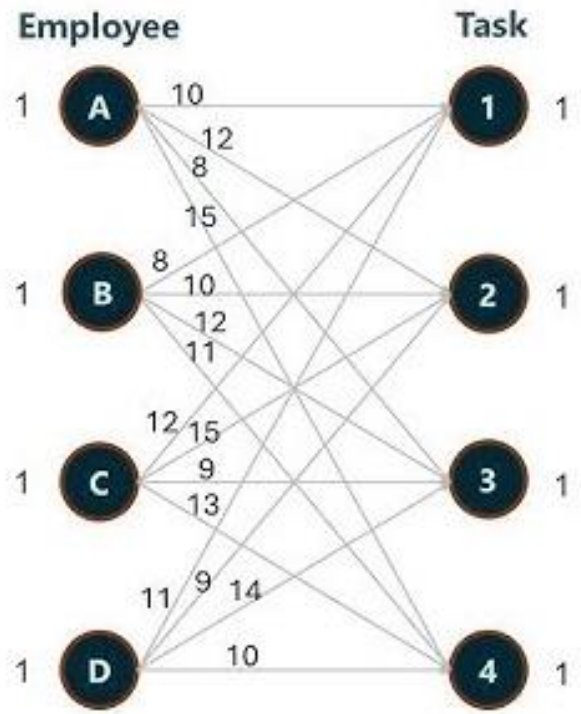
Donc pour éviter ça, il faut que: $x_{11} + x_{12} + x_{13} + x_{21} + x_{22} + x_{23} + x_{31} + x_{32} + x_{33} \le 2$

Remarque: $2 \le |S| \le n-1$ interdit **seulement** les *sous*-tournées qui ne couvrent pas toutes les villes, tout en laissant possible l'unique cycle couvrant l'ensemble entier.

Problème d'affectation(Assignment Problem)

On veut affecter n agents à n tâches. Chaque affectation $\boldsymbol{i \rightarrow j}$ a un coût $\boldsymbol{c_{ij}}$.
Objectif : minimiser le coût total d'affectation en respectant que chaque agent fasse une seule tâche et chaque tâche soit réalisée par un seul agent..

- $\boldsymbol{x_{ij}}$ **= 1** si l'agent i est affecté à la tâche j, 0 sinon.



Formulation en PLNE

$$\begin{cases} min \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} \\ \sum_{i=1}^{n} x_{ij} = 1\,, \forall i \quad \leftarrow \text{Chaque agent est affecté à exactement une tache} \\ \sum_{j=1}^{n} x_{ij} = 1, \forall j \quad \leftarrow \text{Chaque tache est affecté à exactement un agent.} \\ x_{ij} \in \{0,1\} \end{cases}$$

Solutions optimales à valeurs entières

Soit (P) sous forme standard :

$$(P)\begin{cases} Z_{max} = c^T x \\ Ax = \mathrm{b} \\ x \geq 0 \end{cases}$$

où A matrice m x n à coefficients entiers. Les vecteurs c et b sont entiers.
En général, la solution optimale de (P) n'est pas entière.



On cherche une condition suffisante sur la matrice A pour que la solution optimale soit **entière**.

Solutions optimales à valeurs entières

Définition

Une matrice A de taille m x n est dite **totalement unimodulaire** si toute sous-matrice carrée de A à un déterminant qui vaut 0, +1 ou -1

Une sous-matrice est une matrice obtenue à partir d'une matrice en ne gardant que certaines lignes ou colonnes.
Remarque. Toute matrice totalement unimodulaire **est nécessairement** composée de **0, +1 ou -1.**

Théorème

Soit A une matrice **totalement unimodulaire** et **b** et **c** des vecteurs entiers.
Si (P) admet une solution optimale, il admet une solution optimale **entière**.

Remarque. Le résultat est encore vrai si (PL) est écrit sous forme canonique pure avec des contraintes inégalités de la forme $Ax \leq b$ où A est totalement unimodulaire



Comment savoir si une matrice est totalement unimodulaire ?

Proposition (condition suffisante)

Soit **A** une matrice contenant seulement les éléments **0**, **+1** ou **-1** et qui satisfait les 2 conditions suivantes:

1. Chaque colonne contient au plus 2 éléments non-nuls
2. Les lignes de A peuvent être partitionnées en 2 sous-ensembles I_1 et I_2 tels que pour chaque colonne contenant 2 éléments non-nuls :
   - si les 2 éléments non-nuls ont le même signe alors l'un est dans I_1 et l'autre dans I_2.
   - si les 2 éléments non-nuls sont de signes différents alors ils sont tous les deux dans I1 ou tous les deux dans I2.

Solutions optimales à valeurs entières

Quelques exemples

Soit le PL (P1) suivant:

$$\begin{cases} Z_{max} = 2x_1 + 3x_2 + x_3 + 4x_4 + 2x_5 \\ x_1 + x_2 + x_5 \leq 1 \\ -x_1 - x_3 + x_5 \leq 1 \\ x_2 + x_4 \leq 1 \\ -x_3 - x_4 \leq 1 \\ x_1, x_2, x_3, x_4, x_5 \geq 0 \end{cases} \longrightarrow A = \begin{pmatrix} 1 & 1 & 0 & 0 & 1 \\ -1 & 0 & -1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & -1 & -1 & 0 \end{pmatrix}$$

On remarque que: $\det \begin{pmatrix} 1 & 1 \\ -1 & 1 \end{pmatrix} = 2$

Donc A n'est pas totalement unimodulaire (**TUM**)

Soit le PL (P2) suivant:

$$\begin{cases} Z_{max} = 2x_1 + 3x_2 + x_3 + 4x_4 + 2x_5 \\ x_1 + x_2 + x_5 \leq 2 \\ -x_1 - x_3 - x_5 \leq 4 \\ -x_3 - x_4 \leq 1 \\ -x_2 - x_4 \leq 6 \\ x_1, x_2, x_3, x_4, x_5 \geq 0 \end{cases} \longrightarrow A = \begin{pmatrix} 1 & 1 & 0 & 0 & 1 \\ -1 & 0 & -1 & 0 & -1 \\ 0 & 0 & -1 & -1 & 0 \\ 0 & -1 & 0 & -1 & 0 \end{pmatrix}$$

On remarque que chaque colonne contient au plus 2 élément non nul.

Pour I_1 = (ligne 1, ligne 2, ligne 4); I_2 = (ligne 3), on a dans une colonne: 2 éléments ≠ 0 avec le même signe sont dans des matrices différentes; 2 éléments ≠ 0 avec des signes différents sont dans la même matrice ⇒ **A est TUM**

Soit le PL (P3) suivant:

$$\begin{cases} Z_{max} = 2x_1 + 3x_2 + x_3 \\ x_1 - x_2 \leq 8 \\ -x_1 + x_3 \leq 6 \\ x_2 - x_3 \leq 2 \\ x_1, x_2, x_3, \geq 0 \end{cases} \longrightarrow A = \begin{pmatrix} 1 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & -1 \end{pmatrix}$$

On peut facilement vérifier que le déterminant de toute s/matrice carrée = 0 ou -1 ou 1 ⇒ **A est TUM**

Sinon, on peut diviser la matrice A en Pour I_1 = (ligne 1, ligne 2, ligne 3) et I_2 = ∅

Solutions optimales à valeurs entières

Apport des matrices totalement unimodulaires

- En Programmation Linéaire en Nombres Entiers (PLNE), on impose souvent $x_i \in \{0,1\}$ ou $x_i \in \mathbb{Z}$.
- On appelle **relaxation du PLNE** le problème obtenu en remplaçant la contrainte $x_i \in \mathbb{Z}$ (ou $x_i \in \{0, 1\}$) par $x_i \in \mathbb{R}$.
- La relaxation est donc un **programme linéaire (PL)** plus simple à résoudre.
- En général, la solution optimale de la relaxation **n'est pas entière.**
- Exception : si la matrice des contraintes est **totalement unimodulaire** et les vecteurs b et c sont entiers, alors la relaxation admet directement une solution optimale entière (**pas besoin d'utiliser de PLNE** → un simple PL suffit).



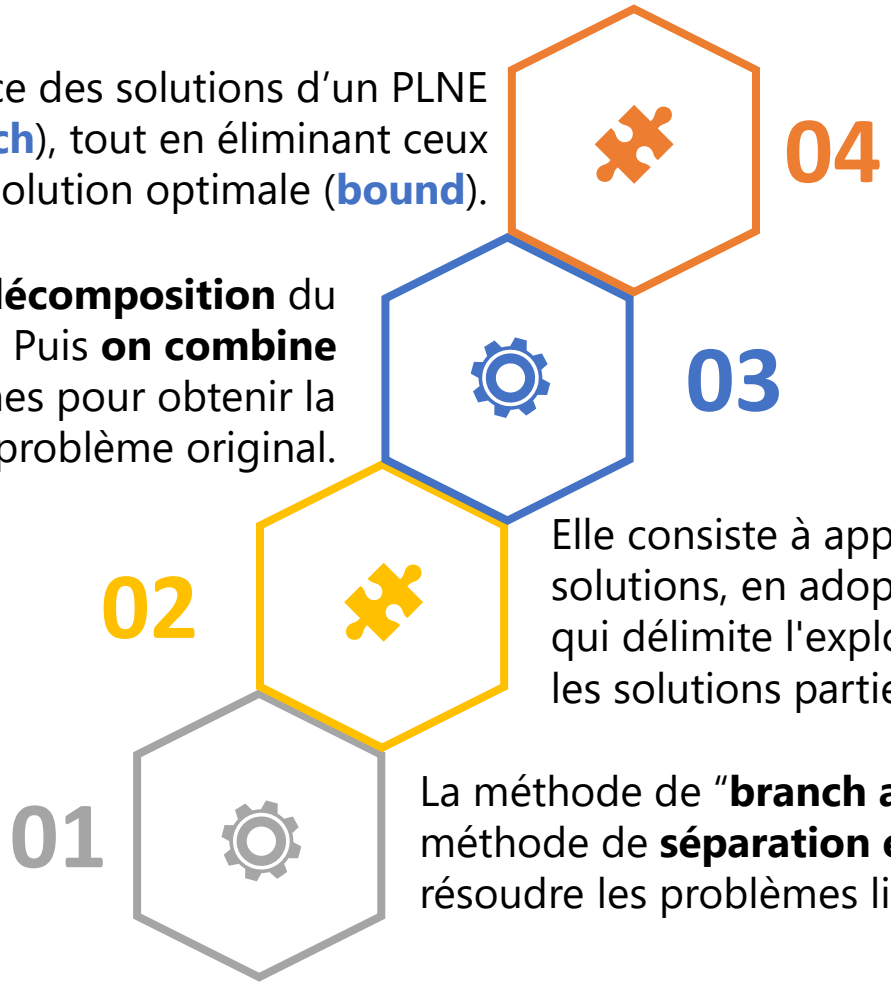
Le théorème n'est pas fait pour « tous les PLNE », mais il est **fondamental** pour **certaines classes de problèmes très courantes** (affectation, transport, etc.), où il explique pourquoi ces problèmes sont « faciles » à résoudre par PL simple, alors que d'autres PLNE (comme sac à dos, TSP) sont difficiles à résoudre.

Méthode de Branch and Bound

Idée Générale

De façon générale, il s'agit d'explorer l'espace des solutions d'un PLNE en le découpant en sous-problèmes (**branch**), tout en éliminant ceux qui ne peuvent pas contenir de solution optimale (**bound**).

Il s'agit essentiellement d'effectuer une **décomposition** du problème en **sous problèmes** plus simples. Puis **on combine** la résolution de tous les sous problèmes pour obtenir la solution du problème original.



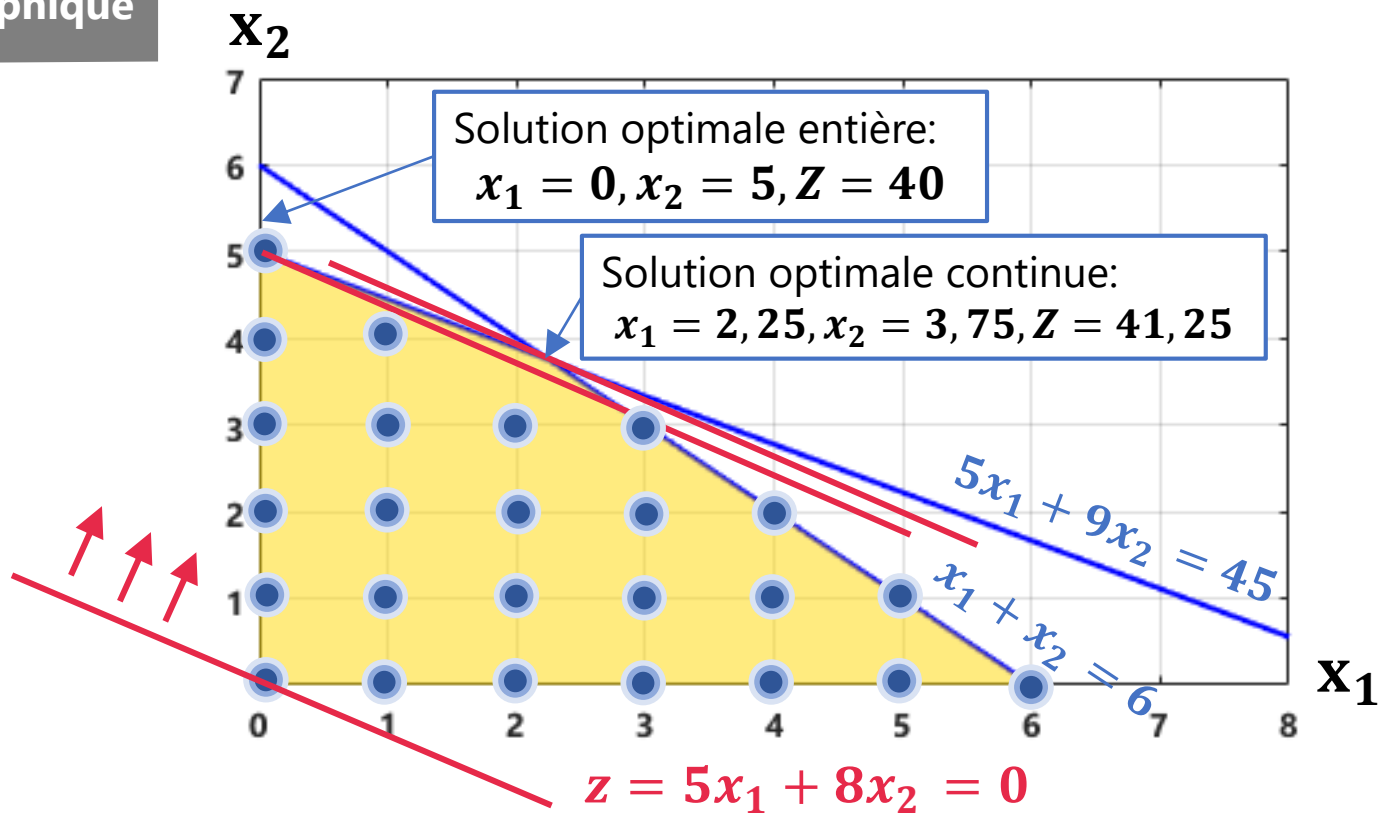
Elle consiste à appliquer une énumération de l'espace des solutions, en adoptant le principe de "**Diviser pour régner**" qui délimite l'exploration de toutes les branches en éliminant les solutions partielles ne menant pas à la solution optimale.

La méthode de "**branch and bound**", encore appelée méthode de **séparation et évaluation** est destinée à résoudre les problèmes linéaires en nombre entier.

Le problème à résoudre | Résolution graphique

Soit le PLNE suivant:

$$(P)\begin{cases} Z_{max} = 5x_1 + 8x_2 \\ x_1 + x_2 \leq 6 \\ 5x_1 + 9x_2 \leq 45 \\ x_1, x_2 \geq 0, \text{entiers} \end{cases}$$



Le domaine réalisable de (P) est constitué des points entiers (bleus) situés dans le polyèdre ;

La solution optimale de (P) est le dernier point Rencontré par une droite de coût constant (rouge) : c'est le point de coordonnées (**0; 5**) de valeur **40**.

Mais cette résolution graphique n'est possible que pour 2 variables) on utilise une méthode arborescente (branch and bound).

On note ($\bar{P}$) la relaxation continue de (P) : il s'agit du problème obtenu à partir de (P) en oubliant (relâchant) la contrainte d'intégrité sur les variables x_1 et x_2 :

$$(\bar{P}) \begin{cases} Z_{max} = 5x_1 + 8x_2 \\ x_1 + x_2 \leq 6 \\ 5x_1 + 9x_2 \leq 45 \\ x_1, x_2 \geq 0 \end{cases}$$

Si (P) admet des solutions (c'est le cas pour cette exemple) et si on désigne par max(P) et max($\bar{P}$) les valeurs optimales de (P) et ($\bar{P}$), alors on a :

max(P) ≤ max($\bar{P}$).

Dans l'exemple précédent, la solution optimale entière = 40 < 41,25, la solution optimale continue.

$\max(P) \leq \max(\bar{P})$

En effet, la solution optimale de (P) n'est qu'une solution admissible de ($\bar{P}$), l'inverse étant le plus souvent faut.

De façon plus générale encore, si l'on considère les deux problèmes suivants, sans faire d'hypothèse sur la fonction f , et liés par une relation d'inclusion de leurs domaines :
($\bar{P}$) : $max_{x \in X}$ f (x) et (P) : $max_{x \in Y \subset X}$ f (x)

Si (P) admet des solutions, on a encore : $\max(P) \leq \max(\bar{P})$:
en élargissant le domaine (P), il devient possible de trouver des solutions meilleures.

Sachant que : $\max(P) \leq \max(\bar{P})$

Corollaire

Si la solution optimale x^* de la relaxation continue $(\bar{P})$ de (P) est entière, alors c'est également la solution optimale de (P) et dans ce cas :

$$\max(P) = \max(\bar{P})$$

- En effet, x^* est alors admissible pour (P) et toutes les solutions admissibles de (P) sont de valeur inférieure ou égale $\max(\bar{P})$ qui est la valeur de x^*.

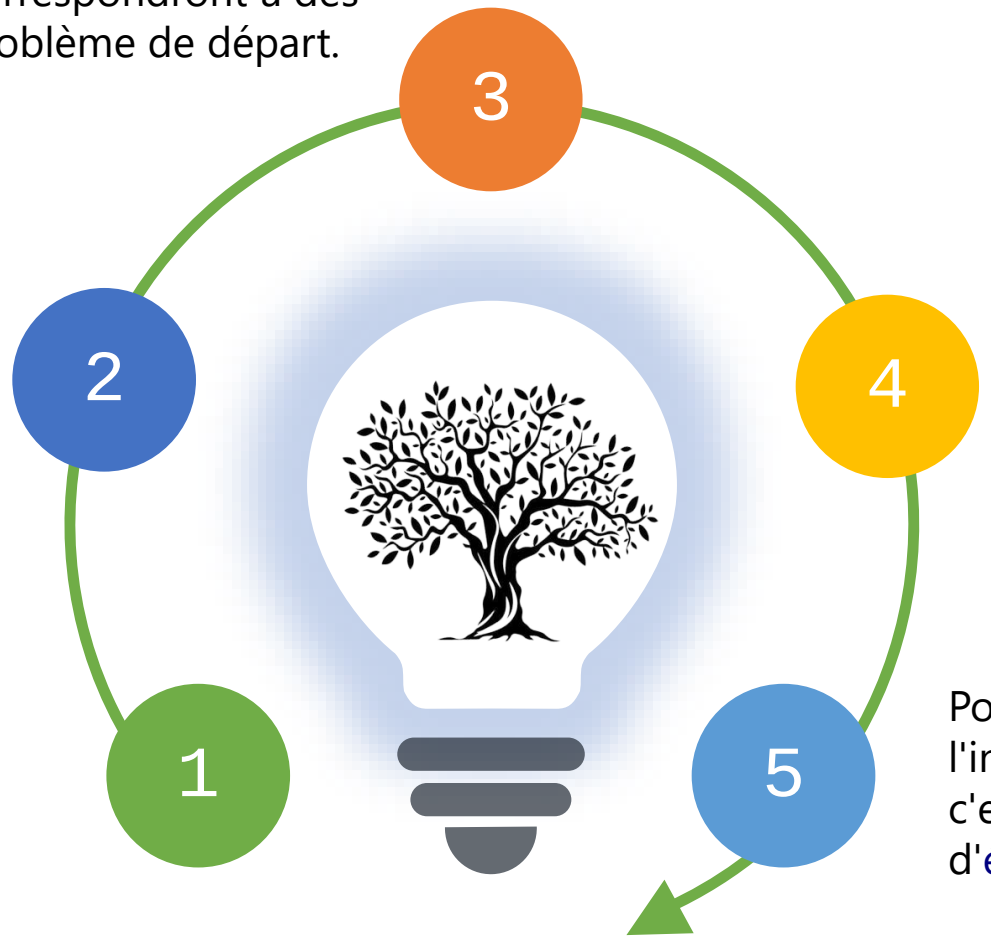
Algorithme de Branch and Bound

Aperçu Général

Les nœuds suivants correspondront à des sous-problèmes du problème de départ.

La racine de l'arbre correspond au problème (P) de départ.

On recherche la solution optimale de (P) en développant un arbre (binaire) de recherche.



À chaque nœud de l'arbre, on calcule la relaxation continue du sous-problème considéré. Cette relaxation continue fournit un majorant (borne) de la solution optimale entière du sous-problème considéré.

Pour éviter de devoir développer l'intégralité de l'arbre, on applique, lorsque c'est possible, trois critères permettant d'élaguer le nœud considéré.

Critères d'élagage

On peut élaguer un nœud lorsque :

1. La solution optimale de la relaxation continue du nœud est entière : on connaît alors la solution optimale entière du nœud, et il n'est pas nécessaire de développer ses nœuds fils;
2. Le sous-problème considéré n'admet aucune solution réalisable (continue)
3. La borne fournie par la relaxation continue est de valeur inférieure ou égale à une solution entière déjà connue du problème (P) de départ : il n'existe pas de possibilité d'améliorer cette solution dans le nœud considéré.



Élaguer un nœud ne signifie **pas nécessairement** que l'optimum du problème de départ ne s'y trouve pas. Cela signifie qu'il n'est pas nécessaire de développer ses nœuds fis, dans la mesure où l'on connaît toute l'information sur la meilleure solution entière de ce nœud.

Algorithme de Branch and Bound

Initialisation

On représente les informations relatives à chaque nœud de l'arbre en reportant :

- Le numéro du problème (problème 0, s/problème 1, etc.),
- La solution optimale de la relaxation continue;
- Sa valeur;
- et on en déduit un majorant de la solution optimale du sous-problème en cours de résolution (si comme dans l'exemple les coeff. dans Z sont tous entiers, on peut prendre comme borne la partie entière inférieure de la relaxation continue)

Problème 0	
$x_1 = 2,25,$ $x_2 = 3,75,$ $Z = 41,25$	**max(P)** $\leq$ **41**

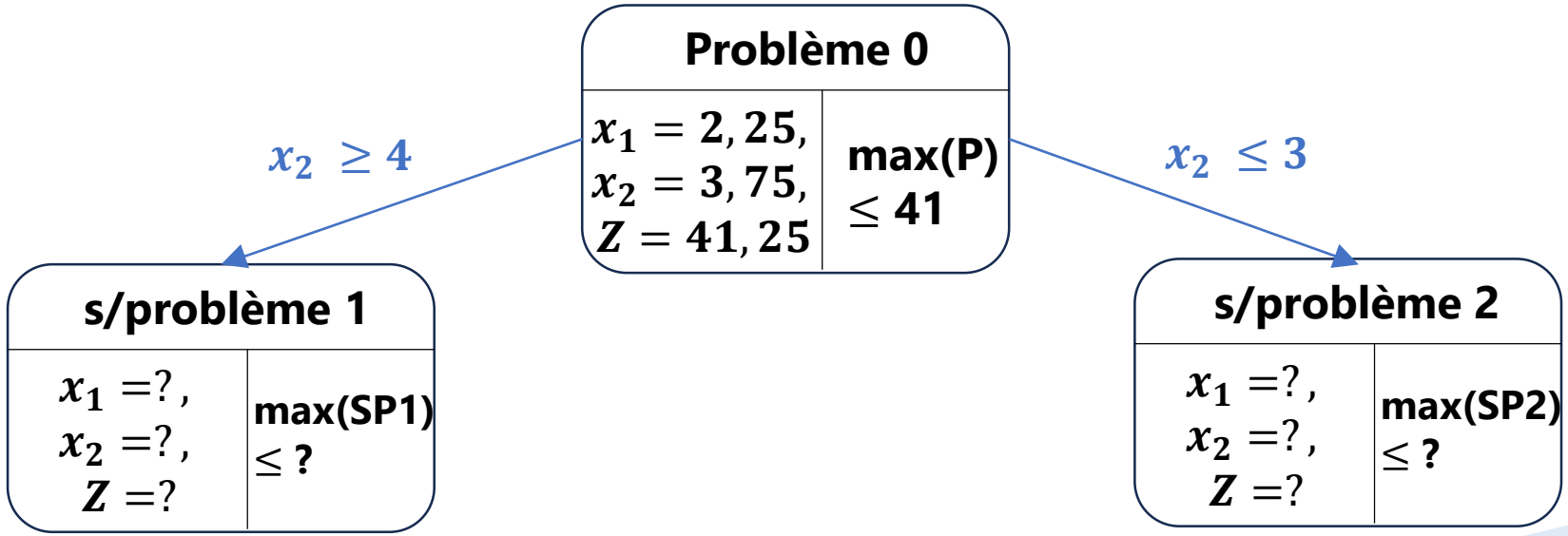
La solution de la relaxation continue n'étant pas entière, on développe deux nœuds fils correspondant aux **s/problèmes 1** et **2**

Algorithme de Branch and Bound

Séparation 1

1. On choisit une variable fractionnaire dans la solution continue précédente. Ici, on a le choix entre x_1 et x_2. Un critère couramment utilisé consiste à choisir la variable qui a la plus grande partie fractionnaire (**règle de dakin**), ici c'est x_2.

2. On effectue un branchement sur les valeurs entières possibles de x2, qui partitionne le domaine réalisable : $x_2 \geq 4$ ou bien $x_2 \leq 3$



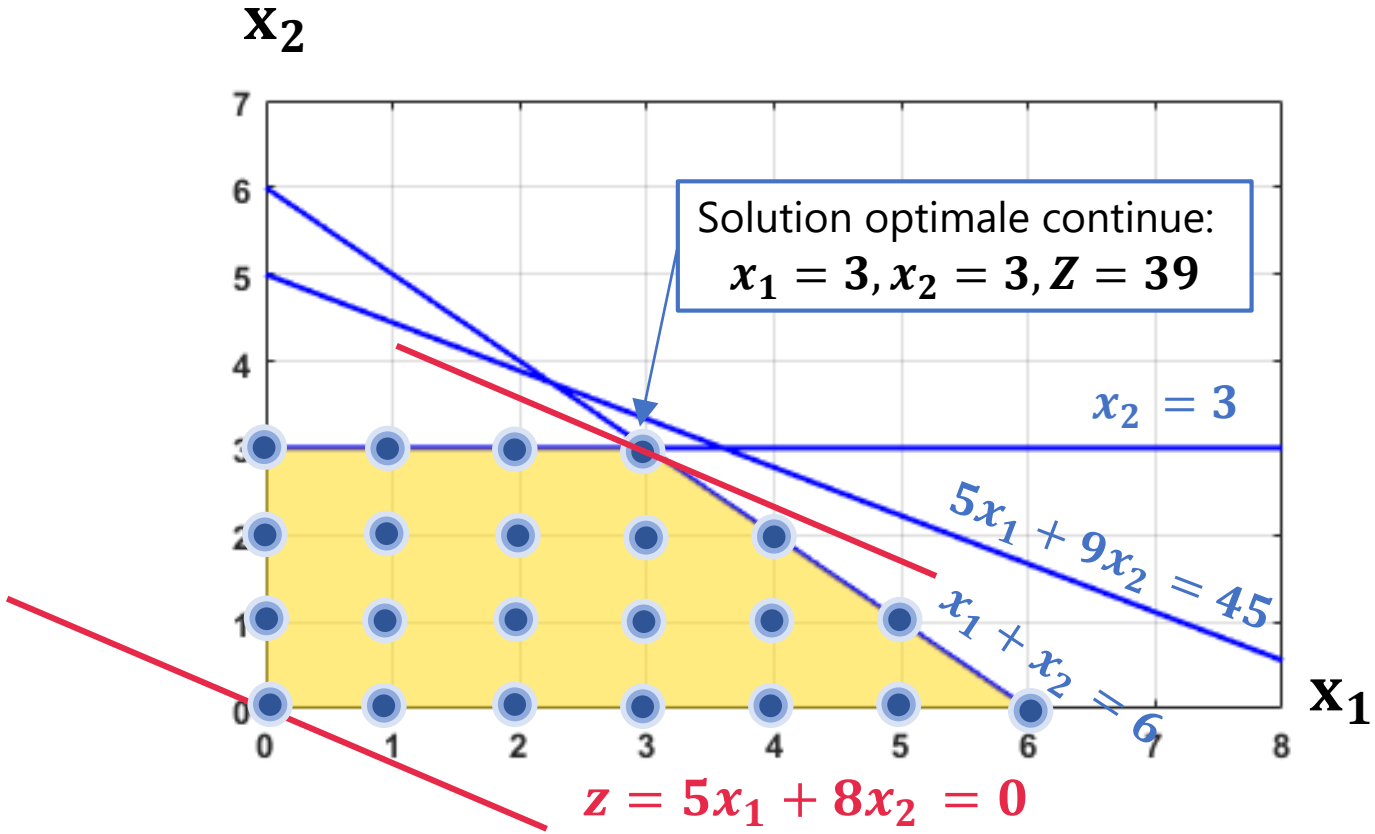
Algorithme de Branch and Bound

Evaluation 1

1. On peut élaguer le nœud du s/problème 2, car la solution optimale de la relaxation continue est entière : on connaît la solution optimale en ce nœud.

2. Élaguer un nœud ne signifie pas que la solution optimale du problème de départ ne s'y trouve pas, cela signifie simplement que l'on dispose de toute l'information nécessaire en ce nœud et qu'il n'est pas nécessaire de développer ses fils.

$$(SP2)\begin{cases} Z_{max} = 5x_1 + 8x_2 \\ x_1 + x_2 \leq 6 \\ 5x_1 + 9x_2 \leq 45 \\ x_2 \leq 3 \\ x_1, x_2 \geq 0 \end{cases}$$

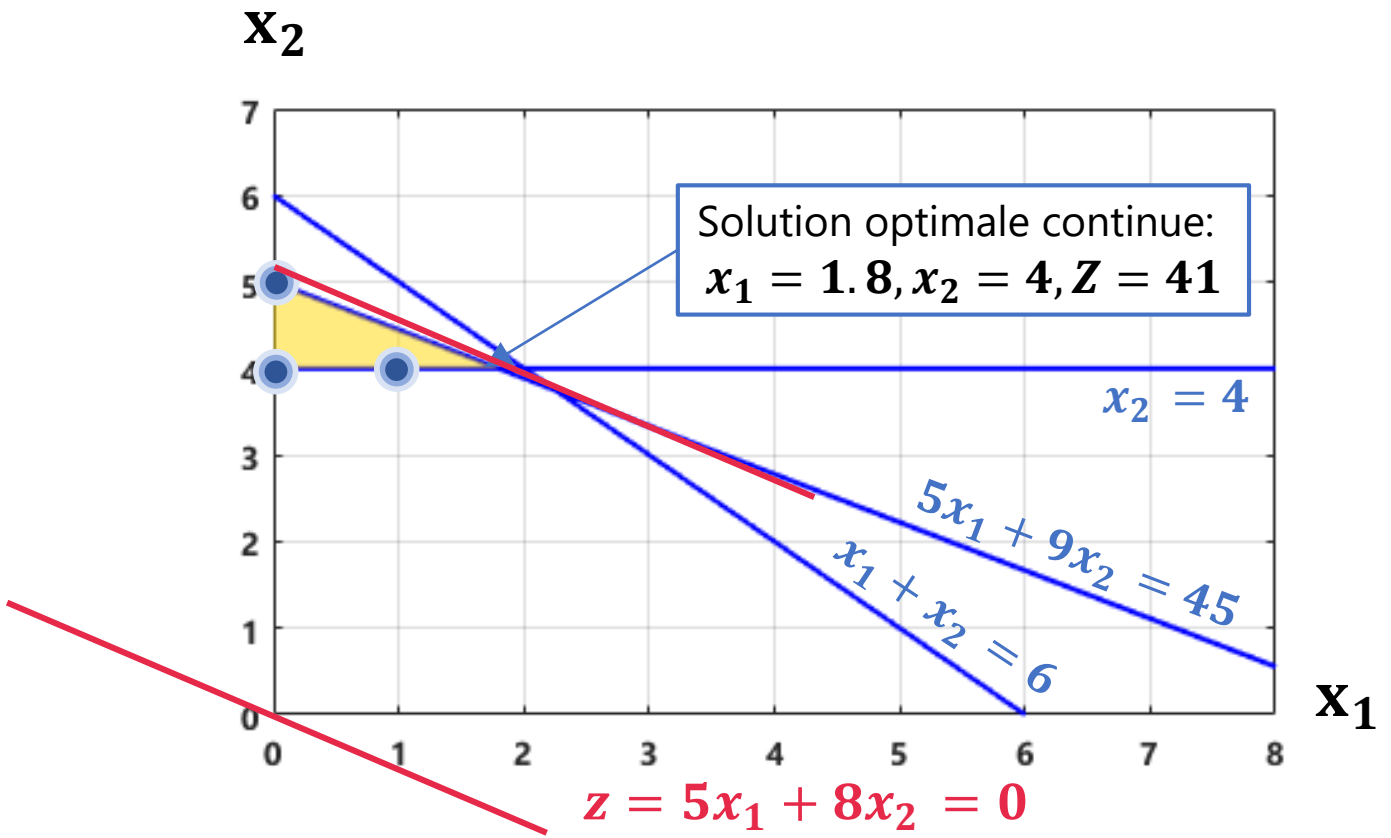


Algorithme de Branch and Bound

Evaluation 2

D'après la solution obtenue pour le **sous-problème 1**, on remarque que la valeur de la variable x_2 est entière, par contre, celle de x_1 est toujours fractionnaire. Donc, on décide de séparer à nouveau sur cette variable, en utilisant la même procédure décrite auparavant.

$$(SP1)\begin{cases} Z_{max} = 5x_1 + 8x_2 \\ x_1 + x_2 \leq 6 \\ 5x_1 + 9x_2 \leq 45 \\ x_2 \geq 4 \\ x_1, x_2 \geq 0 \end{cases}$$



Algorithme de Branch and Bound

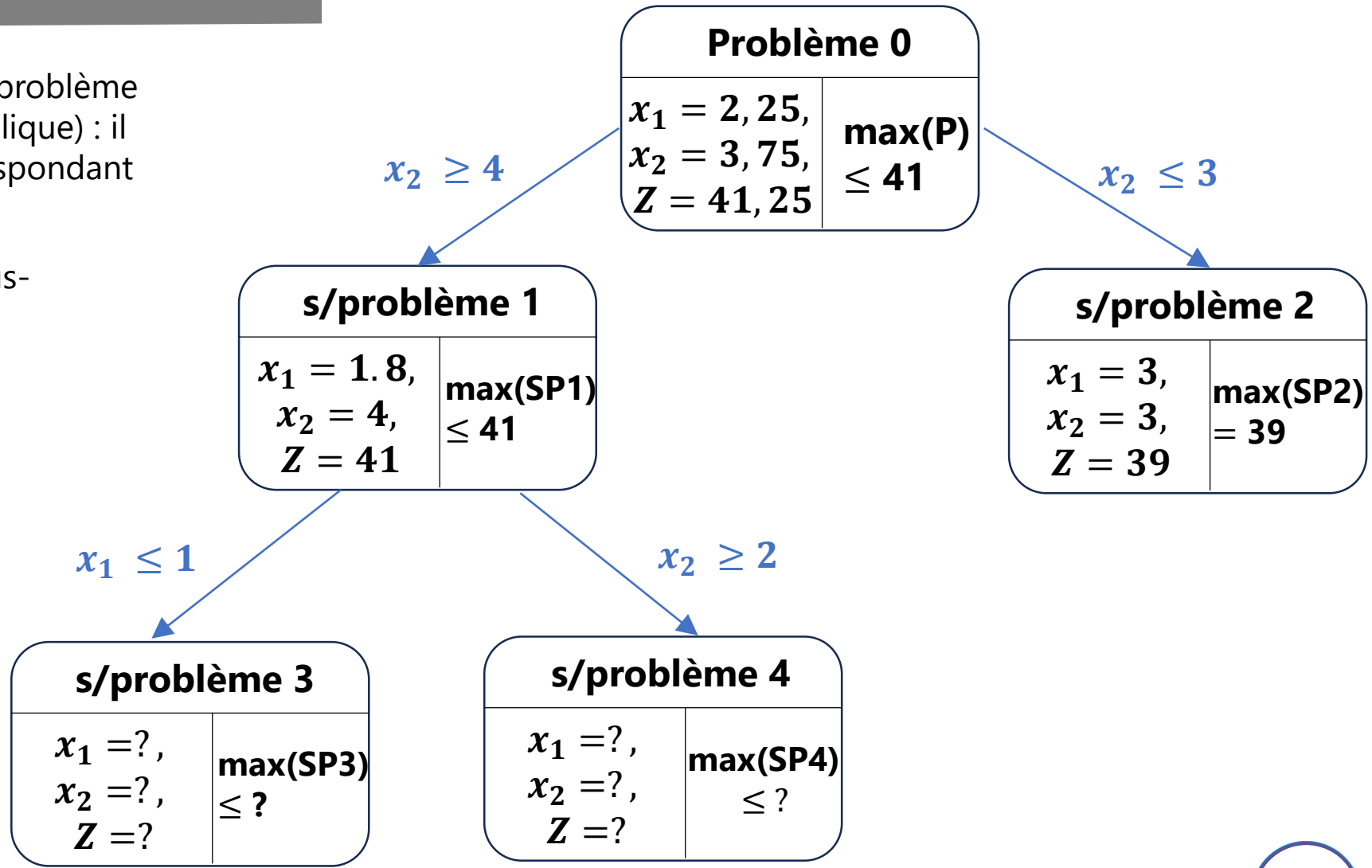
Séparation 2

On ne peut pas élaguer le nœud de sous problème 1 (aucun des 3 critères d'élagage ne s'applique) : il faudra développer deux nœuds fils (correspondant au s/problèmes 3 et 4).

On obtient à nouveau deux sommets (sous-problème 3 et sous problème 4) avec :

$$(SP3)\begin{cases} Z_{max} = 5x_1 + 8x_2 \\ x_1 + x_2 \leq 6 \\ 5x_1 + 9x_2 \leq 45 \\ x_2 \geq 4 \\ x_1 \leq 1 \\ x_1, x_2 \geq 0 \end{cases}$$

$$(SP4)\begin{cases} Z_{max} = 5x_1 + 8x_2 \\ x_1 + x_2 \leq 6 \\ 5x_1 + 9x_2 \leq 45 \\ x_2 \geq 4 \\ x_1 \geq 2 \\ x_1, x_2 \geq 0 \end{cases}$$



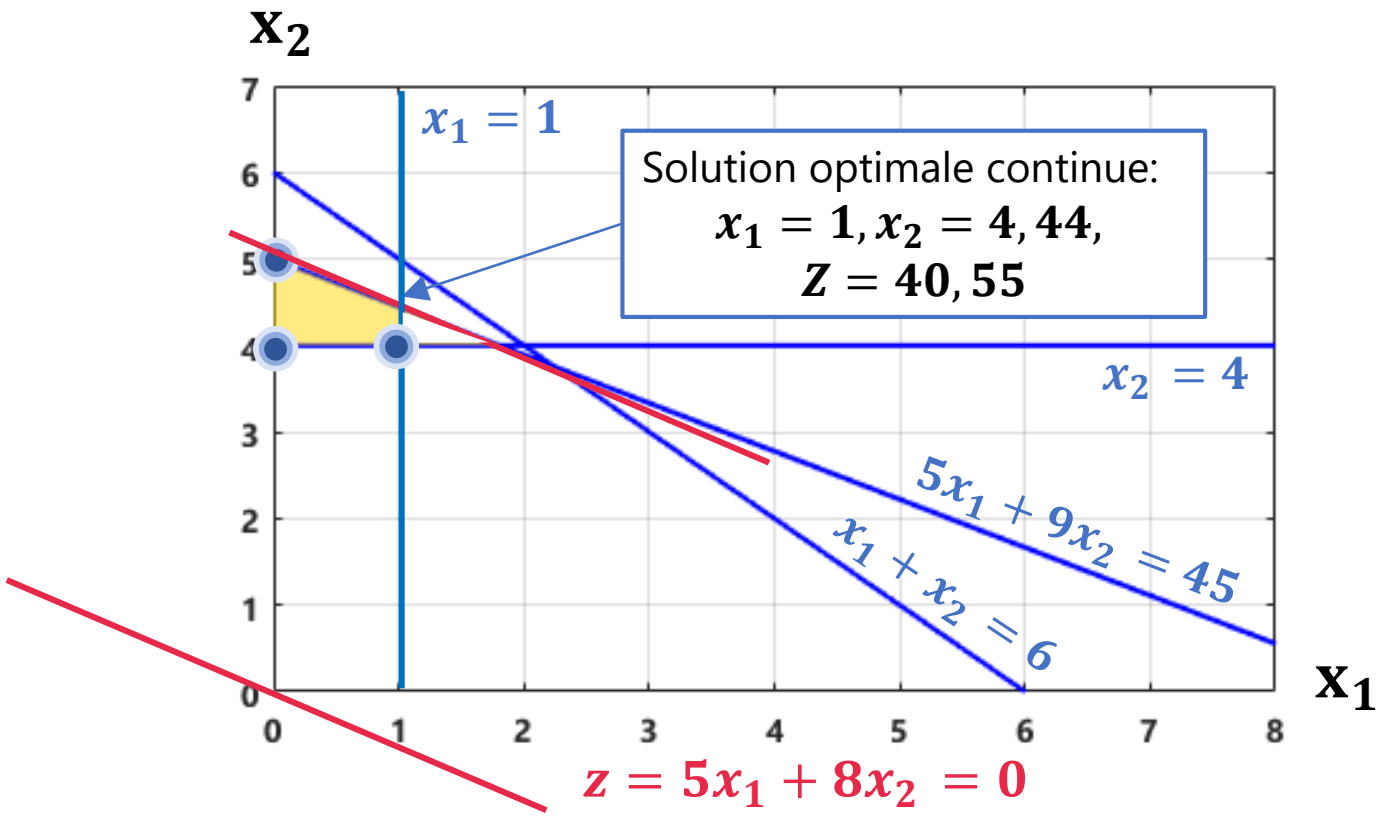
Algorithme de Branch and Bound

Evaluation 3

La solution du sous problème relaxé (SP3) donne la solution x_1 entier et x_2 **fractionnaire**. Donc l'étape suivante consiste à faire un « branch » à se sommet en le divisant en s/problèmes 5 et 6.

On peut très bien constater que le s/problème 4 (avec $x_1 \geq 2$) n'admet pas de solution réalisable.

$$(SP3)\begin{cases} Z_{max} = 5x_1 + 8x_2 \\ x_1 + x_2 \leq 6 \\ 5x_1 + 9x_2 \leq 45 \\ x_2 \geq 4 \\ x_1 \leq 1 \\ x_1, x_2 \geq 0 \end{cases}$$



Algorithme de Branch and Bound

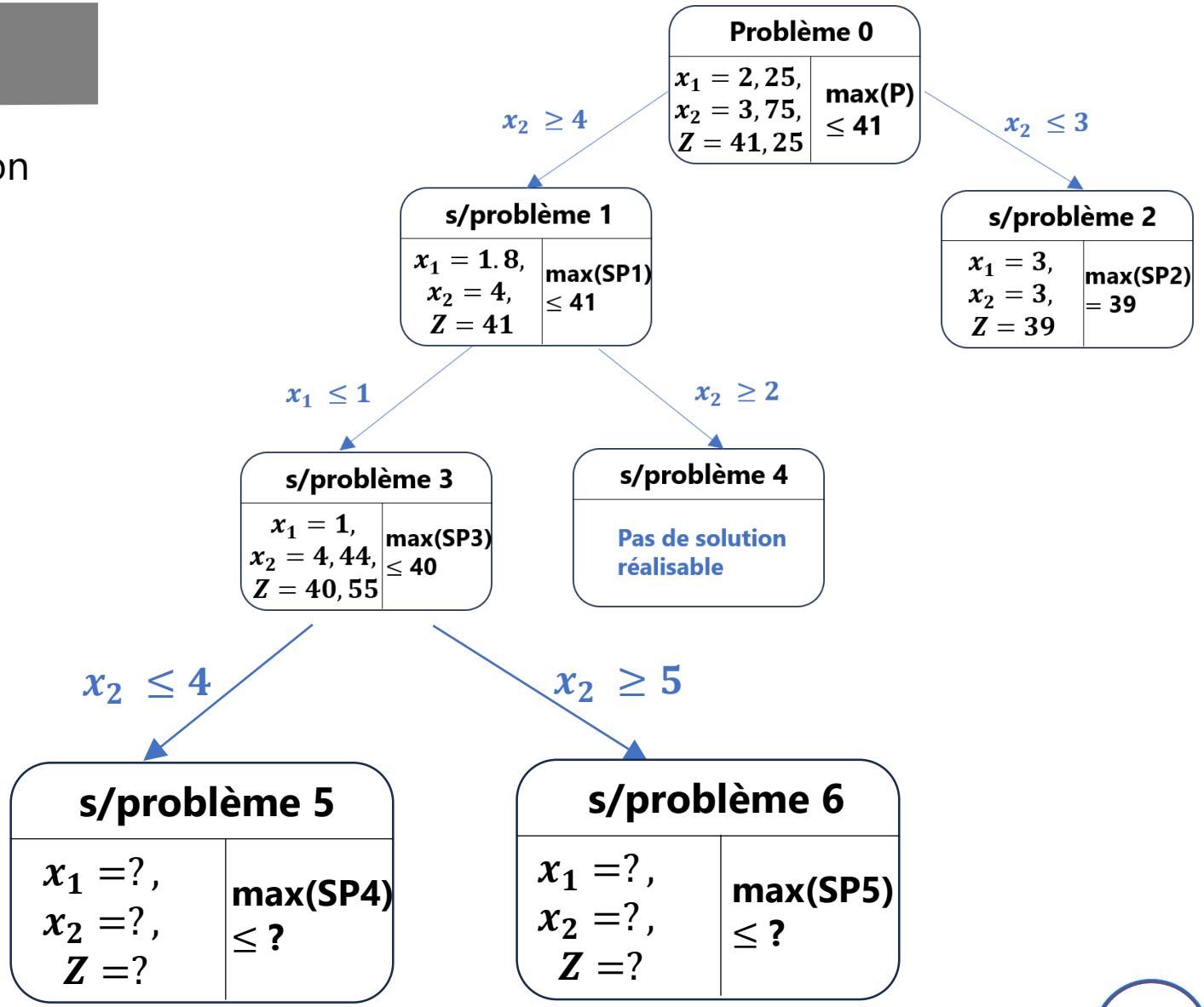
Séparation 3

On choisit une variable fractionnaire dans la solution continue du s/problème 3 : x_2 est la seule variable fractionnaire.

On effectue le branchement sur x_2 : $x_2 \leq 4$ ou bien $x_2 \geq 5$:

$$(SP5)\begin{cases} Z_{max} = 5x_1 + 8x_2 \\ x_1 + x_2 \leq 6 \\ 5x_1 + 9x_2 \leq 45 \\ x_2 \geq 4 \\ x_1 \leq 1 \\ x_2 \leq 4 \\ x_1, x_2 \geq 0 \end{cases}$$

$$(SP6)\begin{cases} Z_{max} = 5x_1 + 8x_2 \\ x_1 + x_2 \leq 6 \\ 5x_1 + 9x_2 \leq 45 \\ x_2 \geq 4 \\ x_1 \leq 1 \\ x_2 \geq 5 \\ x_1, x_2 \geq 0 \end{cases}$$



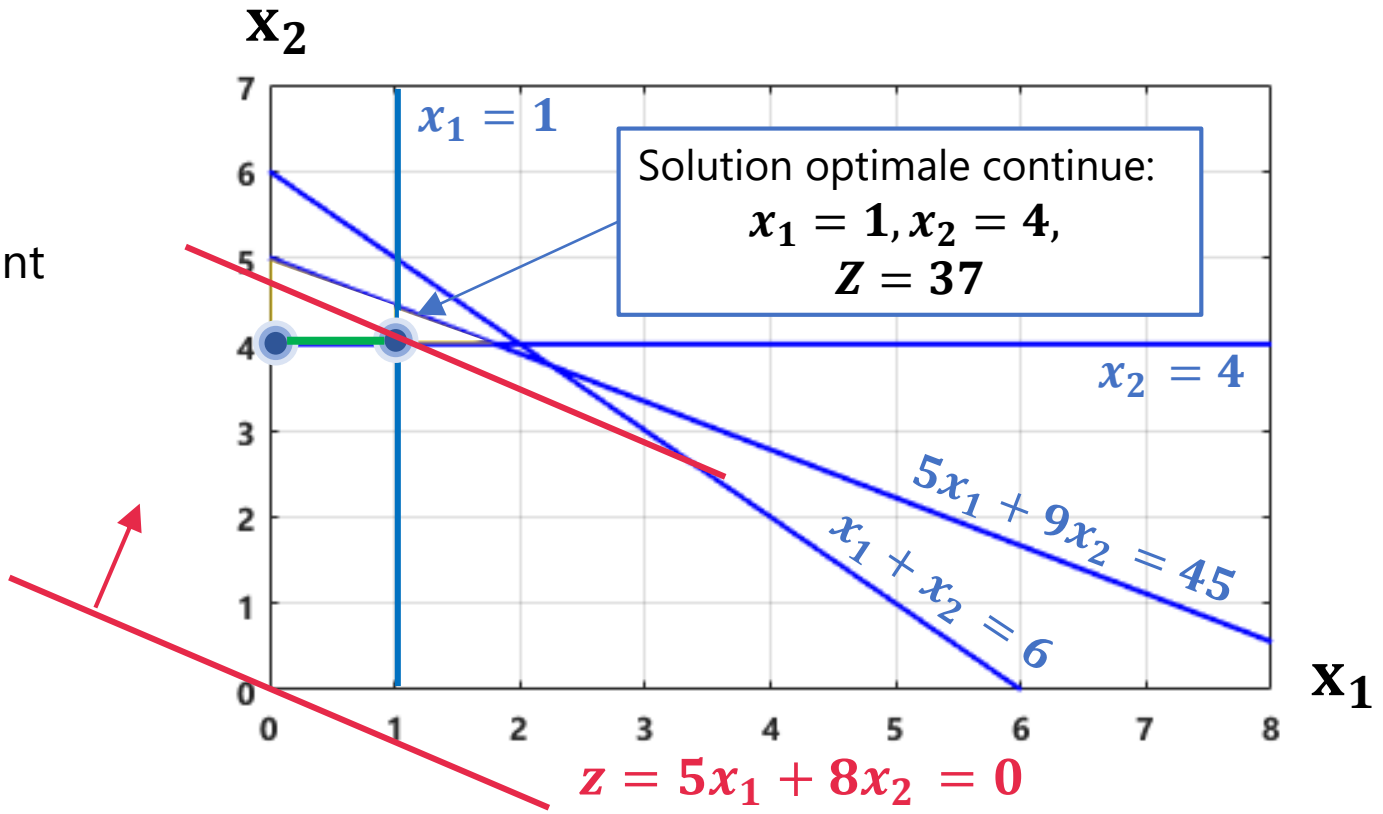
Algorithme de Branch and Bound

Evaluation 4

On recalcule la solution pour les deux nouveaux sommets. On commence par le sommet du sous problème 5.

Nous avons : ($x_2 \leq 4$ et $x_2 \geq 4$) ⇒ ($x_2 = 4$)
Le polyèdre de la relaxation continue est le segment **vert**, d'extrémités (0; 4) et (1; 4).

$$(SP5)\begin{cases} Z_{max} = 5x_1 + 8x_2 \\ x_1 + x_2 \leq 6 \\ 5x_1 + 9x_2 \leq 45 \\ x_2 \geq 4 \\ x_1 \leq 1 \\ x_2 \leq 4 \\ x_1, x_2 \geq 0 \end{cases}$$



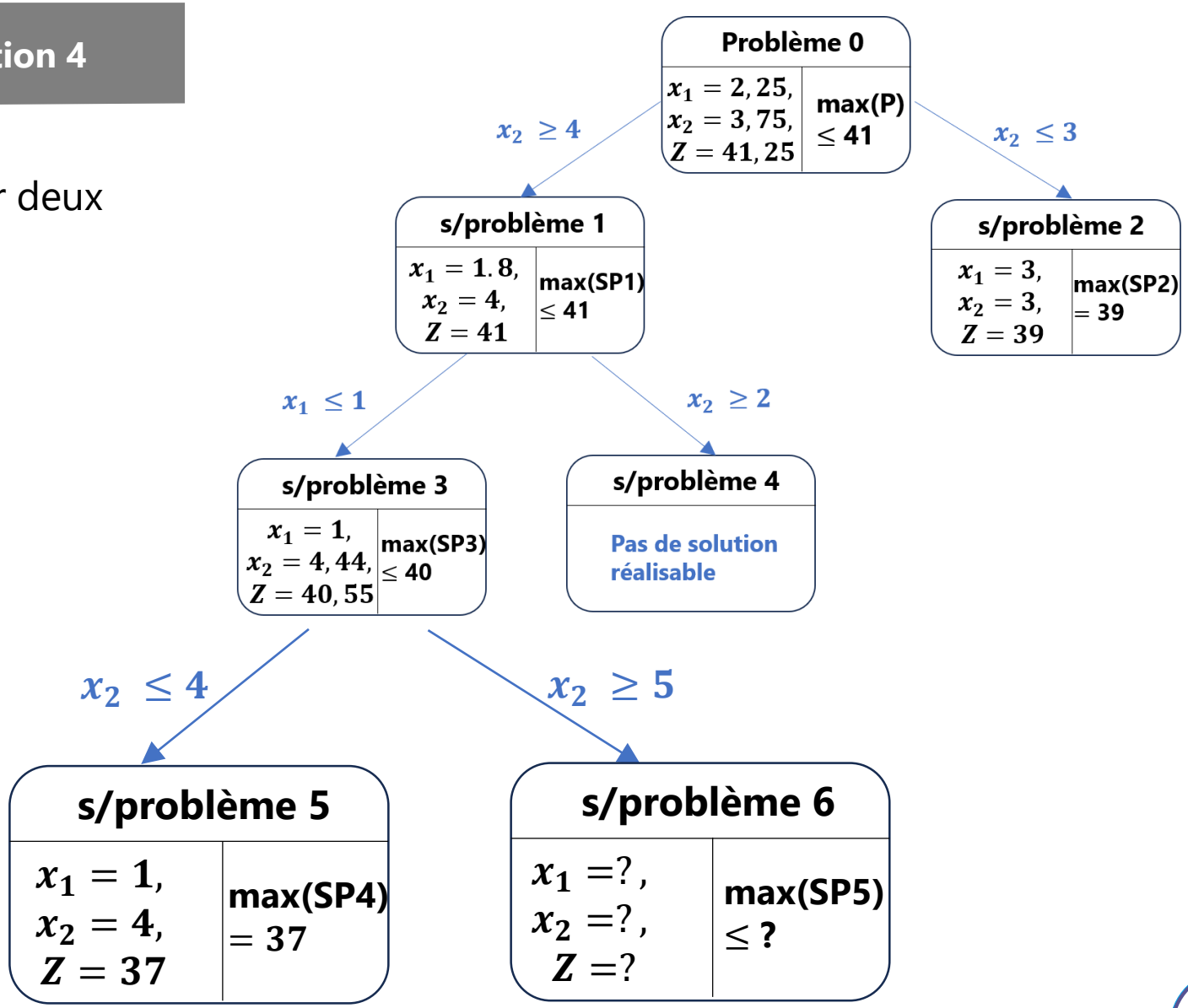
Algorithme de Branch and Bound

Evaluation 4

On peut élaguer le nœud du s/problème 5 pour deux raisons :

La solution de la relaxation continue du s/problème 4, (1; 4) est entière : on connaît la solution optimale relative à ce nœud, donc il n'est pas nécessaire de développer ses fils.

On connaît depuis l'étape 2 une solution entière (3; 3) du problème 2 de départ de valeur 39, meilleure que l'optimum de (SP4) de valeur 37 :



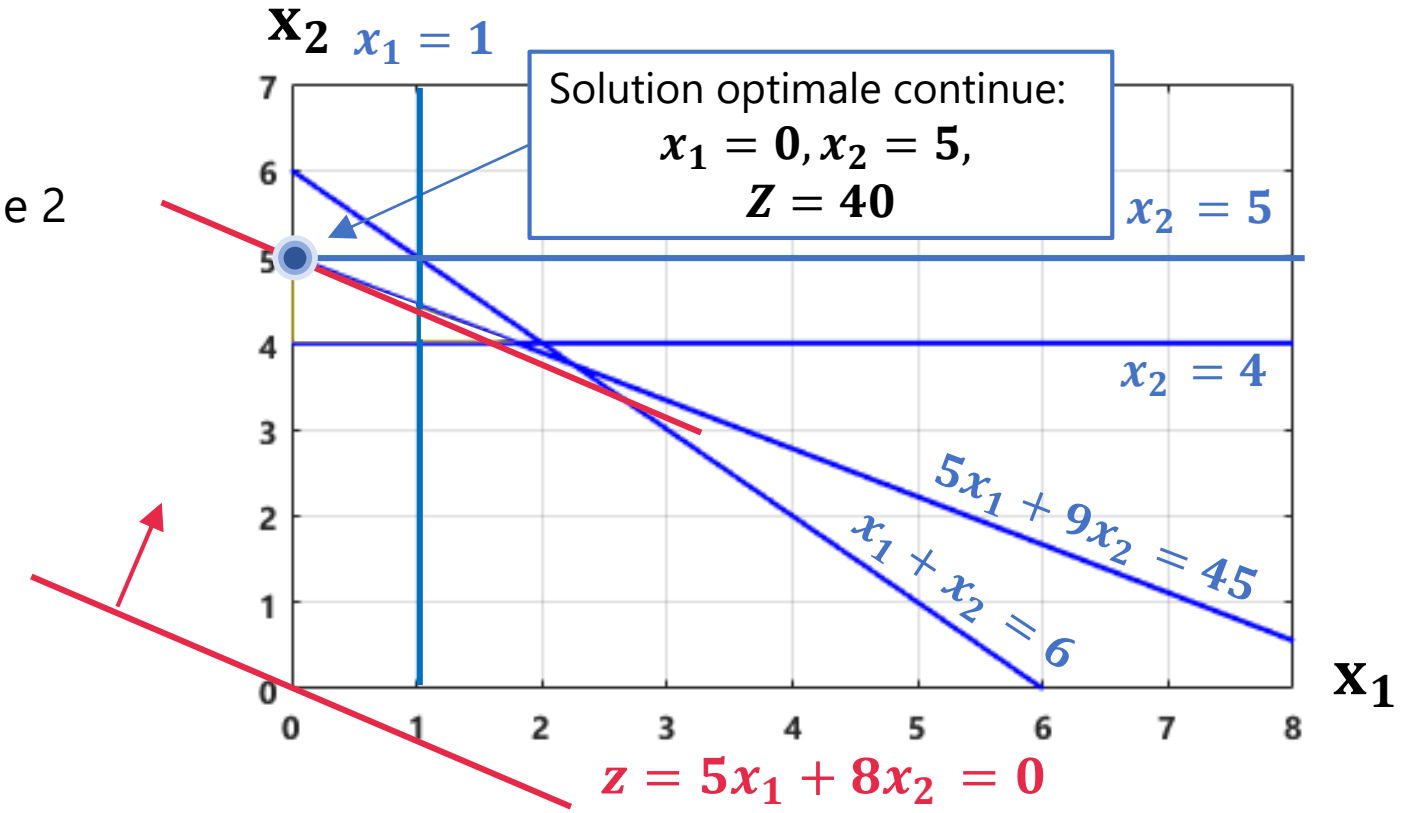
Algorithme de Branch and Bound

Evaluation 4

Le polyèdre de la relaxation continue est ici restreint à un unique point : le point (0; 5).

On remarque La valeur optimale du sous-problème 2 (**z =39**) est inférieure à la valeur optimale donnée par le sous-problème 5 (**z=40**).

$$(SP6)\begin{cases} Z_{max} = 5x_1 + 8x_2 \\ x_1 + x_2 \leq 6 \\ 5x_1 + 9x_2 \leq 45 \\ x_2 \geq 4 \\ x_1 \leq 1 \\ x_2 \geq 5 \\ x_1, x_2 \geq 0 \end{cases}$$



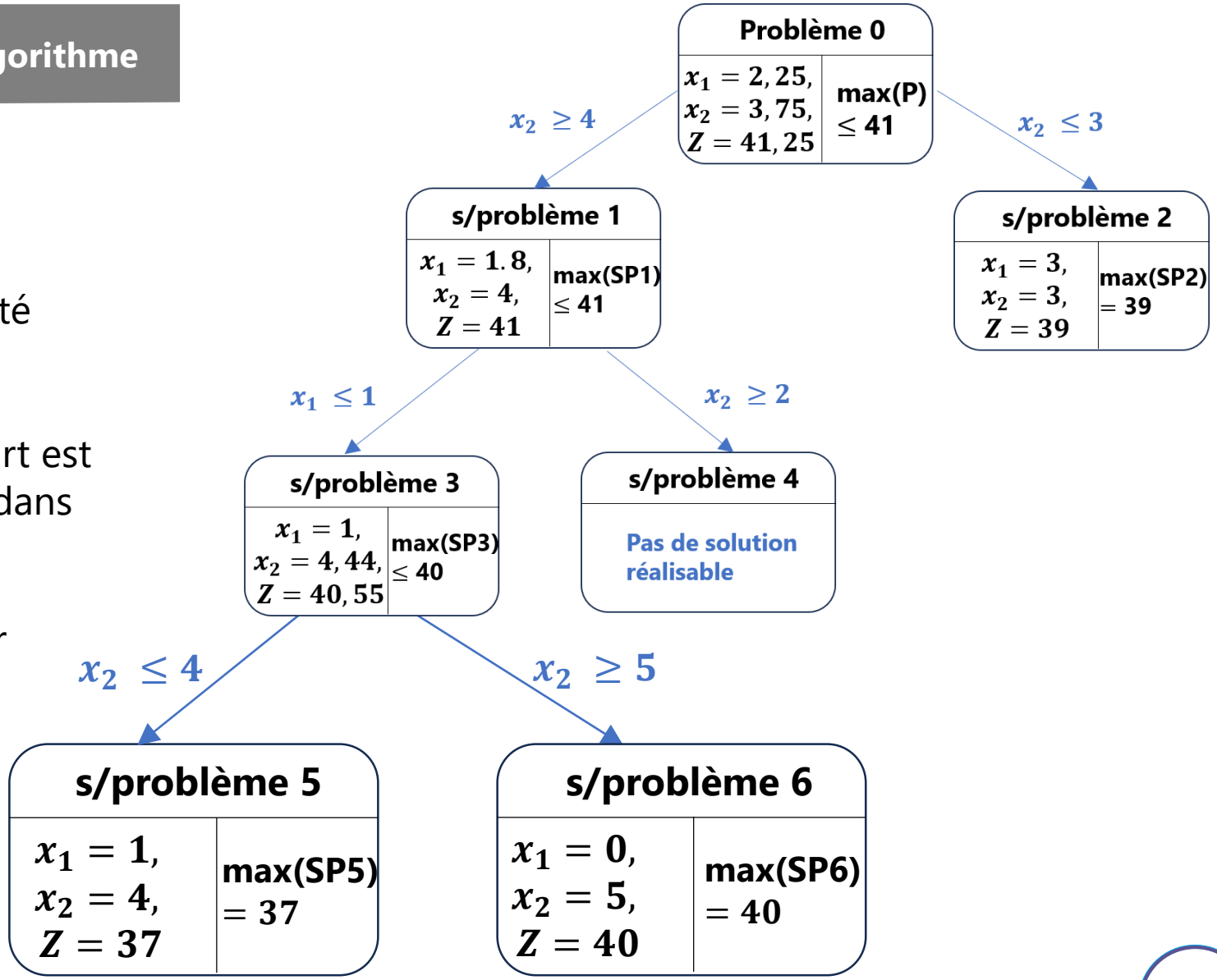
Algorithme de Branch and Bound

Fin de l'algorithme

L'algorithme s'arrête car tous les nœuds ont été élagués.

La solution optimale du problème (P) de départ est alors la meilleure solution entière rencontrée dans l'ensemble des nœuds développés.

C'est donc la solution $x_1 = 0$; $x_2 = 5$, de valeur $Z = 40$, trouvée à l'étape 6.



Algorithme de Branch and Bound

Remarques

Lorsqu'on conçoit un algorithme de branch-and-bound on doit préciser :

- **La stratégie d'exploration :** lorsque plusieurs nœuds peuvent être développés, lequel doit-on développer en premier ? La stratégie la plus couramment utilisée, dite du meilleur d'abord, consiste à choisir le nœud ayant **la plus grande valeur** de relaxation continue (l'intuition étant qu'il a plus de chance de contenir la solution optimale du problème de départ). Il existe d'autres stratégies, comme la stratégie profondeur d'abord.

- **La stratégie de branchement :** lorsqu'on a choisi le nœud à développer, sur quelle variable va-t-on brancher ? Une stratégie consiste à choisir la variable fractionnaire **ayant la plus grande partie fractionnaire**. (Là encore, il existe d'autres stratégies).



Lorsque tous les nœuds ont été élagués, la solution optimale du problème de départ est la meilleure solution entière rencontrée au cours de l'algorithme de branch-and-bound.

Méthode des Plans Sécants (Cutting planes)

Idée de base

1

Résoudre la relaxation linéaire du problème (c'est-à-dire le programme linéaire obtenu en supprimant les contraintes d'intégralité sur les variables).



2

Si la solution du PL est entière, alors elle est **optimale** pour le problème initial (le programme linéaire en nombres entiers).
☞ **Fin de l'algorithme.**

3

Si la solution du PL n'est pas entière, il faut **trouver une nouvelle contrainte linéaire** (appelée **coupe**) qui :

- **exclut** la solution fractionnaire actuelle,
- mais **ne supprime aucune solution entière** valide.
- (Une telle coupe existe toujours.)

4

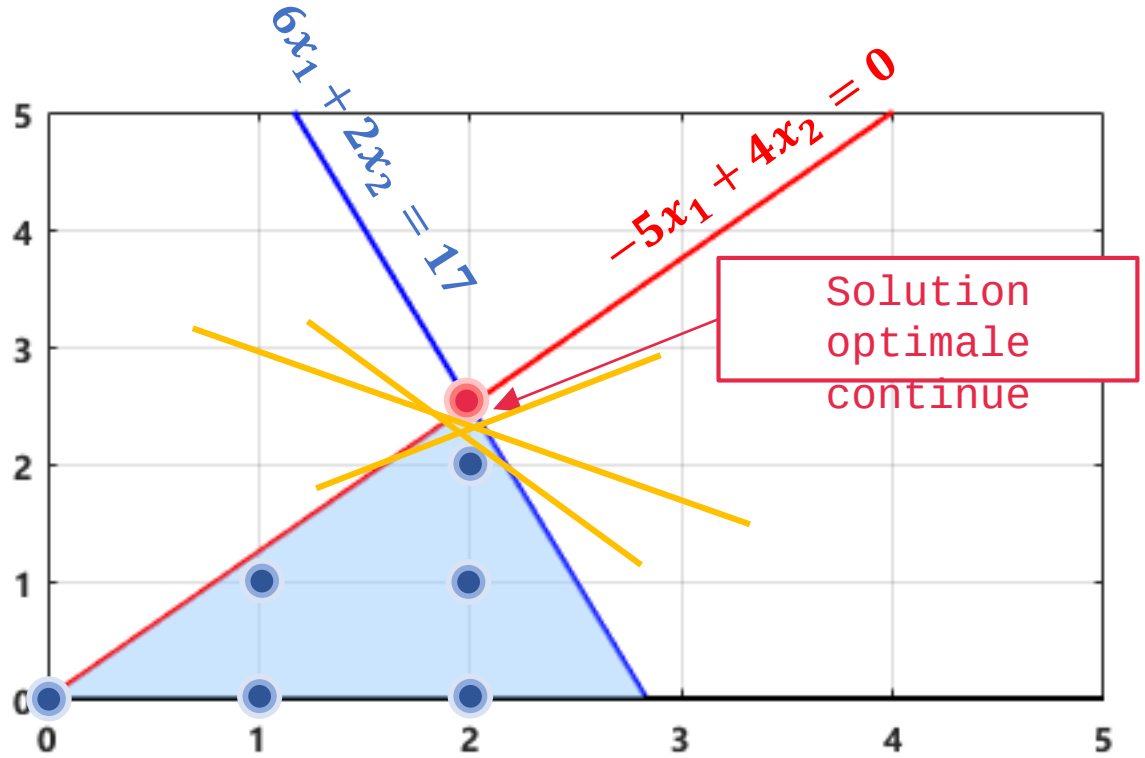
Ajouter cette coupe au problème, puis **revenir à l'étape 1**.
☞ On résout à nouveau la relaxation linéaire enrichie par cette nouvelle contrainte.

Méthode des Plans Sécants (Cutting planes)

Principe

Soit le programme linéaire relaxé suivant:

$$(P)\begin{cases} Z_{max} = x_1 + x_2 \\ -5x_1 + 4x_2 \leq 0 \\ 6x_1 + 2x_2 \leq 17 \\ \textit{Coupes valides} \\ x_1, x_2 \geq 0 \end{cases}$$



Une **coupe** doit à la fois :

- **exclure la solution** obtenue par la relaxation linéaire (la solution fractionnaire),
- tout en **conservant l'ensemble de toutes les solutions entières faisables** (ne pas les éliminer).

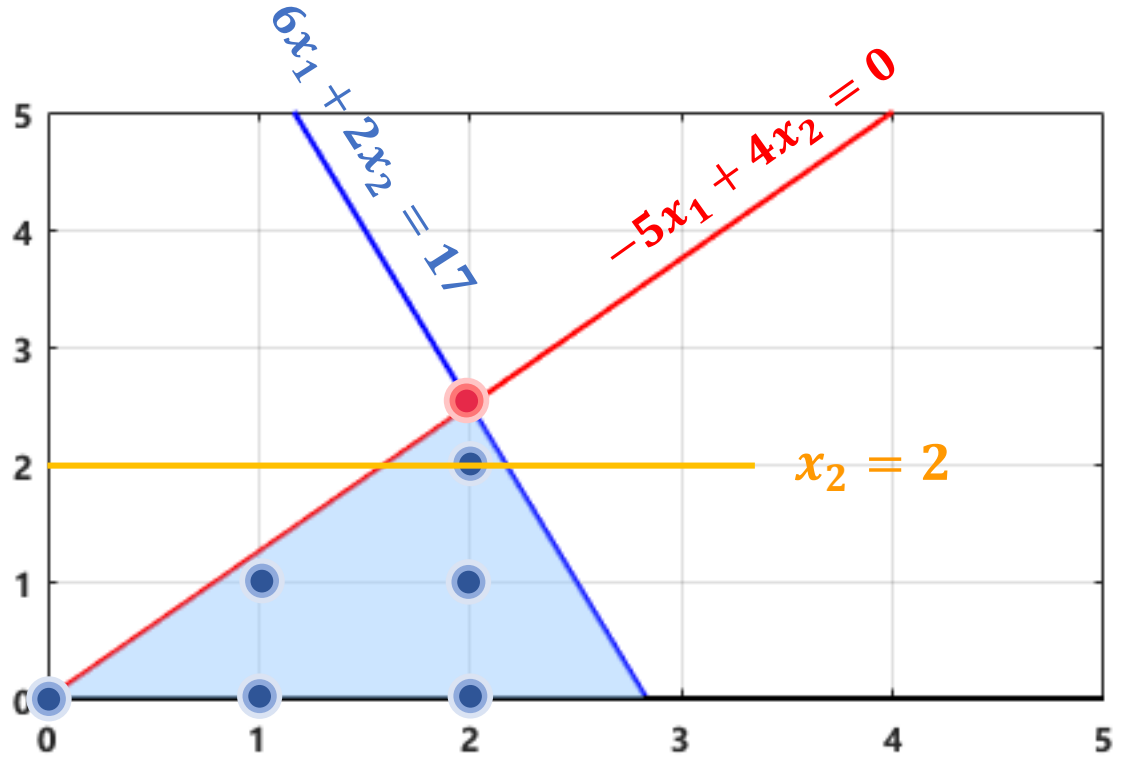
☞ Il existe **toujours au moins une coupe valide**

Méthode des Plans Sécants (Cutting planes)

Principe

Soit le programme linéaire relaxé suivant:

$$(P)\begin{cases} Z_{max} = x_1 + x_2 \\ -5x_1 + 4x_2 \leq 0 \\ 6x_1 + 2x_2 \leq 17 \\ x_2 \leq 2 \\ x_1, x_2 \geq 0 \end{cases}$$



La contrainte $x_2 \leq 2$ est une **coupe valide**, car elle **exclut la solution optimale** du programme linéaire relaxé, **sans éliminer aucun point entier.**

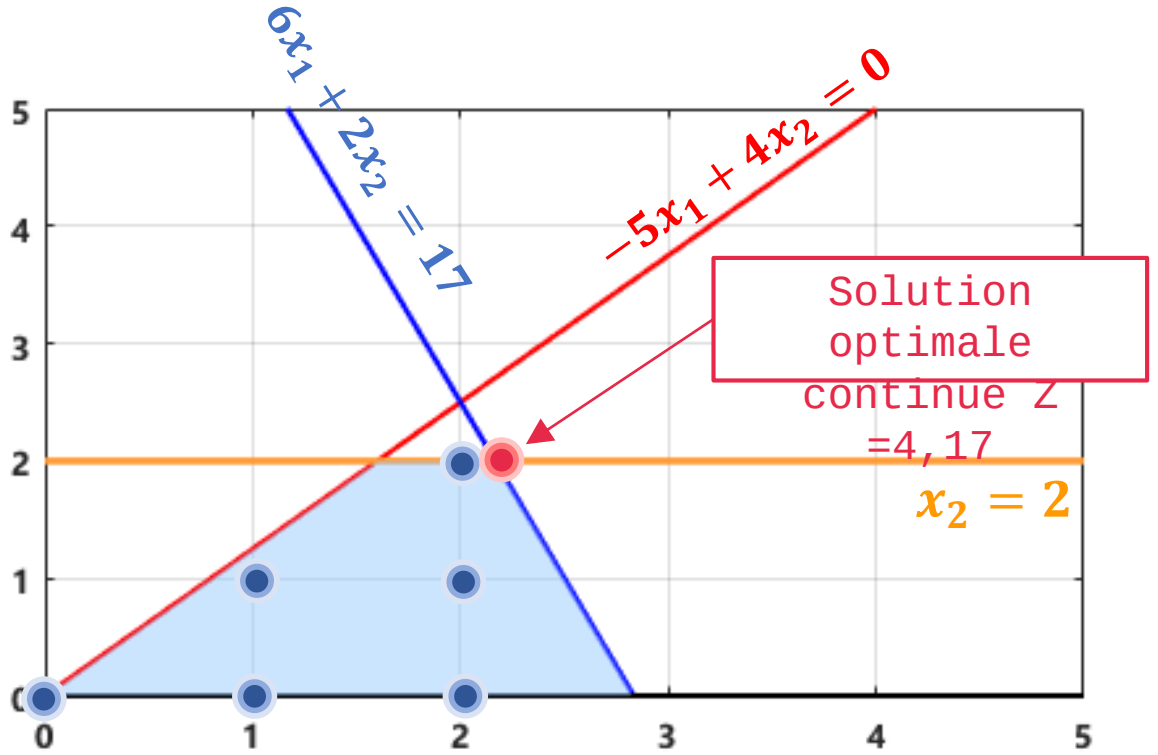
☞ On résout maintenant **la relaxation linéaire** pour ce **nouveau problème** (avec la coupe ajoutée)...

Méthode des Plans Sécants (Cutting planes)

Principe

Soit le programme linéaire relaxé suivant:

$$(P)\begin{cases} Z_{max} = x_1 + x_2 \\ -5x_1 + 4x_2 \leq 0 \\ 6x_1 + 2x_2 \leq 17 \\ x_2 \leq 2 \\ x_1, x_2 \geq 0 \end{cases}$$



L'ajout d'une **coupe** réduit notre **borne supérieure**, car il réduit l'ensemble des solutions faisables (on a ajouté une nouvelle contrainte).

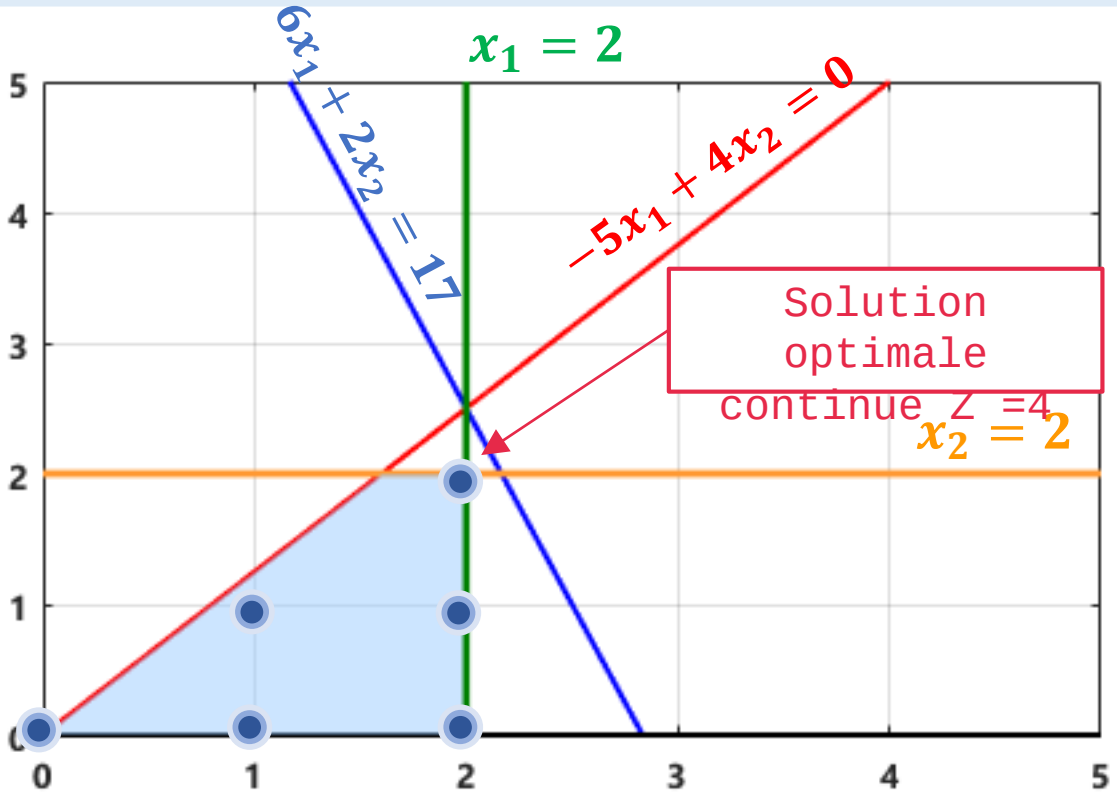
☞ La solution **n'est toujours pas entière**, il faut donc **ajouter une autre coupe !**

Méthode des Plans Sécants (Cutting planes)

Principe

Soit le programme linéaire relaxé suivant:

$$(P)\begin{cases} Z_{max} = x_1 + x_2 \\ -5x_1 + 4x_2 \leq 0 \\ 6x_1 + 2x_2 \leq 17 \\ x_2 \leq 2 \\ x_1 \leq 2 \\ x_1, x_2 \geq 0 \end{cases}$$



La **solution du programme linéaire est entière**, elle est donc **également optimale pour le problème en nombres entiers d'origine.**

Méthode des Plans Sécants (Cutting planes)

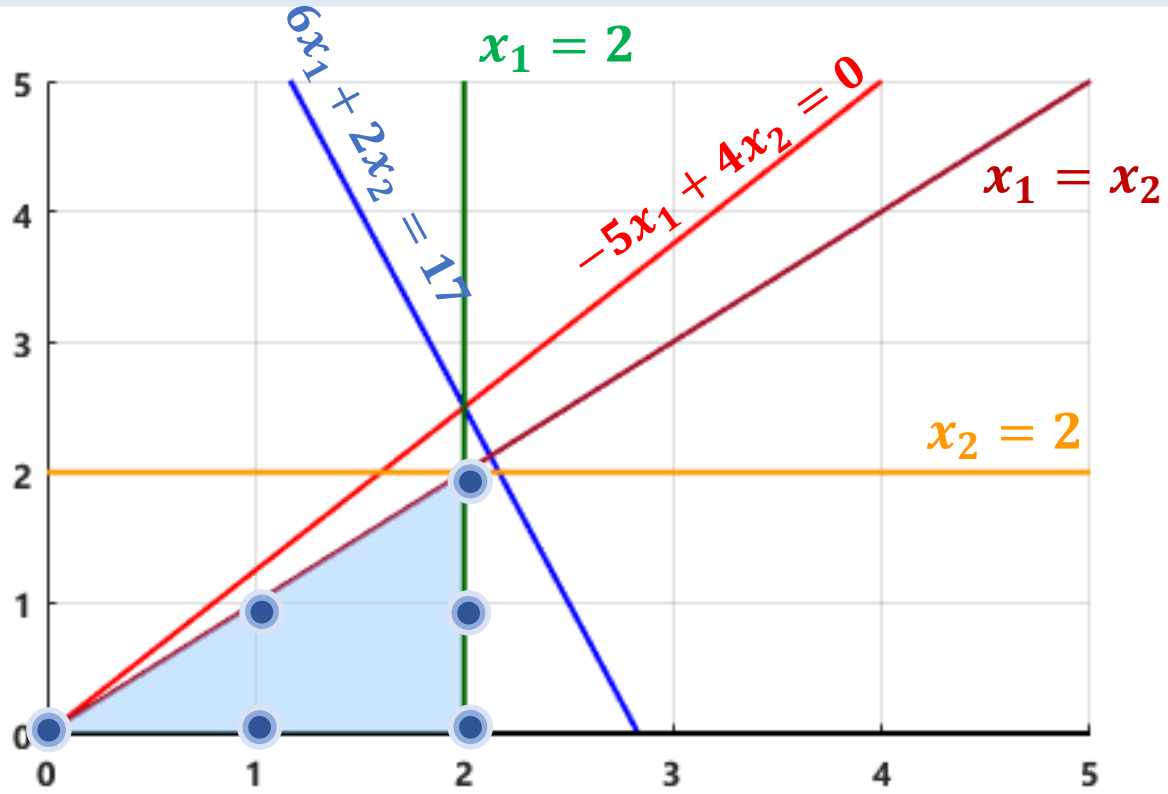
Principe

Si l'on ajoute **toutes les contraintes linéaires possibles** qui **n'excluent aucun point entier faisable**, on obtient **l'enveloppe convexe** de l'ensemble des points entiers faisables.

☞ Si l'on utilise cette **enveloppe convexe**, alors la **relaxation linéaire** fournit **toujours la véritable solution optimale** du problème.

L'**enveloppe convexe** est en général **très difficile à calculer** lorsqu'on a un **grand nombre de variables de décision**.

☞ En utilisant une méthode des plans sécants (cutting plane), on peut (espérons-le) **trouver le point optimal** sans avoir à calculer **toute l'enveloppe convexe**.



$$(P)\begin{cases} Z_{max} = x_1 + x_2 \\ -5x_1 + 4x_2 \leq 0 \\ 6x_1 + 2x_2 \leq 17 \\ x_2 \leq 2 \\ x_1 \leq 2 \\ x_1 \leq x_2 \\ x_1, x_2 \geq 0 \end{cases}$$

Principe



Nous savons que chaque **coupe** doit :

- **Éliminer** la solution fractionnaire actuelle du programme linéaire ;
- **Préserver** toutes les solutions entières faisables.

☞ Mais comment construire une telle coupe en pratique ?

Coupes de Gomory (Gomory Cuts)

Idée Générale

Une méthode célèbre pour créer des coupes valides est appelée **coupe de Gomory**, découverte par le mathématicien américain **Ralph Gomory** en **1950.**

Méthode **opérationnelle** (basée sur les **tableaux du simplexe**).

Après avoir résolu le **relaxé linéaire** d'un PLNE, on **analyse les lignes du tableau du simplexe** pour identifier une **variable de base fractionnaire.**

A partir de cette ligne, on **construit une inégalité coupante** (appelée coupe de Gomory) qui :

- élimine la solution fractionnaire actuelle,
- mais conserve **tous les points entiers faisables.**

Coupes de Gomory (Gomory Cuts) | Règle Générale

Théorème

Supposons que les variables entières non négatives $x_1, x_2, \ldots, x_n$ satisfont l'équation

$$a_1x_1 + a_2x_2 + \cdots + a_nx_n = b.$$

Alors, elles satisfont également l'inégalité

$$\lfloor a_1 \rfloor x_1 + \lfloor a_2 \rfloor x_2 + \cdots + \lfloor a_n \rfloor x_n \leq \lfloor b \rfloor,$$

où $\lfloor r \rfloor$ désigne la **partie entière de** r, c'est-à-dire **le plus grand entier inférieur ou égal à** r. De plus, la différence entre les deux membres de cette inégalité est un **entier**.

⚠ **Attention lors de la prise de la partie fractionnaire des nombres négatifs !**

Si r est un nombre réel **positif**, alors $r - \lfloor r \rfloor$ représente simplement la partie de r après la virgule. si $r = 1{,}23$ alors : $r - \lfloor r \rfloor = 0{,}23$

Cependant, cette propriété **n'est plus vraie** lorsque r est **négatif**. Par exemple, si $r = -1{,}23$, alors sa partie entière est $\lfloor r \rfloor = -2$, et sa partie fractionnaire vaut : $r - \lfloor r \rfloor = 0{,}77$.

Coupes de Gomory (Gomory Cuts)

Règle Générale

Théorème

Supposons que les variables entières non négatives $x_1, x_2, \ldots, x_n$ satisfont l'équation

$$a_1x_1 + a_2x_2 + \cdots + a_nx_n = b.$$

Alors, elles satisfont également l'inégalité

$$\lfloor a_1 \rfloor x_1 + \lfloor a_2 \rfloor x_2 + \cdots + \lfloor a_n \rfloor x_n \leq \lfloor b \rfloor,$$

où $\lfloor r \rfloor$ désigne la **partie entière de** r, c'est-à-dire **le plus grand entier inférieur ou égal à** r.

✳ **Forme générale d'une coupe de Gomory :**

De plus, la différence entre les deux membres de cette inégalité est un **entier**. $\sum_j (a_{ij} - \lfloor a_{ij} \rfloor)\, x_j \geq b_j - \lfloor b_j \rfloor$

Où :

la ligne ***i*** correspond à la **contrainte liée à la variable de base fractionnaire**.

⚙ **Application :**

1. On ajoute la coupe au système de contraintes.
2. On résout à nouveau le programme linéaire relaxé.
3. On répète le processus jusqu'à obtenir une

Coupes de Gomory (Gomory Cuts)

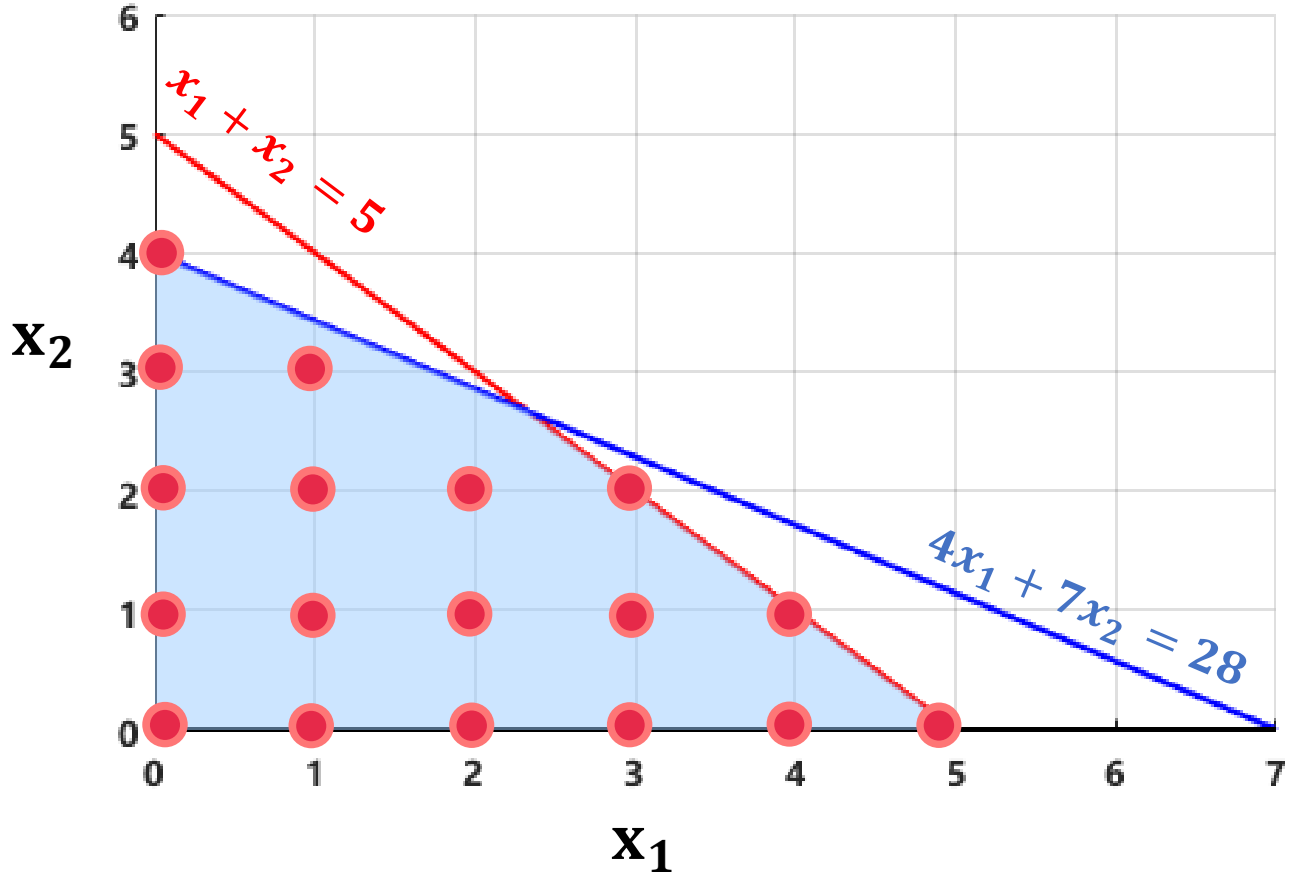
Exemple

Considérant le PLNE suivant:

$$(P)\begin{cases} Z_{max} = 5x_1 + 6x_2 \\ x_1 + x_2 \leq 5 \\ 4x_1 + 7x_2 \leq 28 \\ x_1, x_2 \in \mathbb{N} \end{cases}$$

Le relaxé linéaire sous forme standard:

$$(P)\begin{cases} Z_{max} = 5x_1 + 6x_2 \\ x_1 + x_2 + x_3 = 5 \\ 4x_1 + 7x_2 + x_4 = 28 \\ x_1, x_2 \geq 0 \end{cases}$$



Coupes de Gomory (Gomory Cuts)

Exemple

Le relaxé linéaire sous forme standard:

$$(P)\begin{cases} Z_{max} = 5x_1 + 6x_2 \\ x_1 + x_2 + x_3 = 5 \\ 4x_1 + 7x_2 + x_4 = 28 \\ x_1, x_2 \geq 0 \end{cases}$$

Résolution par tableaux de simplexe ⟶

	x_1	x_2 ↓	x_3	x_4	b
x_3	1	1	1	0	5
← x_4	4	7	0	1	28
-Z	5	6	0	0	0

1ère Itération :

	x_1 ↓	x_2	x_3	x_4	b
← x_3	3/7	0	1	-1/7	1
x_2	4/7	1	0	1/7	4
-Z	11/7	0	0	-6/7	-24

2ème Itération :

	x_1	x_2	x_3	x_4	b
x_1	1	0	7/3	-1/3	7/3
x_2	0	1	-4/3	1/3	8/3
-Z	0	0	-11/3	-1/3	-83/3

Coupes de Gomory (Gomory Cuts)

Exemple

Tableau Optimal:

	x_1	x_2	x_3	x_4	b
x_1	1	0	7/3	-1/3	7/3
x_2	0	1	-4/3	1/3	8/3
-Z	0	0	-11/3	-1/3	-83/3

1 Choisir l'une des **variables de base ayant une valeur fractionnaire** et écrire **son équation** à partir du tableau optimal.

$$x_1 + \frac{7}{3}x_3 - \frac{1}{3}x_4 = \frac{7}{3}$$

2 Réécrire l'équation de manière à exprimer les **valeurs fractionnaires** comme la **somme d'une partie entière** et d'une **partie décimale**.

$$x_1 + \left(\frac{6}{3} + \frac{1}{3}\right)x_3 + \left(-\frac{3}{3} + \frac{2}{3}\right)x_4 = \left(\frac{6}{3} + \frac{1}{3}\right)$$

3 Conserver les **valeurs entières** et les **variables à coefficients entiers** du côté **gauche** de l'équation, et **déplacer les fractions** ainsi que les **variables à valeurs fractionnaires** du côté **droit**.

$$x_1 + 2x_3 - x_4 - 2 = -\frac{1}{3}x_3 - \frac{2}{3}x_4 + \frac{1}{3}$$

Coupes de Gomory (Gomory Cuts)

Exemple

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

Partons de l'équation ci-dessus et Rappelons que les variables x_3 et x_4 sont des entiers, on se retrouve avec:

$$-\frac{1}{3}x_3 - \frac{2}{3}x_4 + \frac{1}{3} \leq 0 \Leftrightarrow -\frac{1}{3}x_3 - \frac{2}{3}x_4 \leq -\frac{1}{3}$$

L'équation de la coupe de **gomory** après standardisation est donnée par:

$$-\frac{1}{3}x_3 - \frac{2}{3}x_4 + x_5 = -\frac{1}{3}$$

Tableau Optimal:

	x_1	x_2	x_3	x_4	b
x_1	1	0	7/3	-1/3	7/3
x_2	0	1	-4/3	1/3	8/3
-Z	0	0	-11/3	-1/3	-83/3

Introduire l'équation de la coupe de gomory dans le tableau optimal

	x_1	x_2	x_3	x_4	x_5	b
x_1	1	0	7/3	-1/3	0	7/3
x_2	0	1	-4/3	1/3	0	8/3
x_5	0	0	1/3	2/3	1	1/3
-Z	0	0	-11/3	-1/3	0	-83/3

Coupes de Gomory (Gomory Cuts)

Exemple

On remarque que le membre droit de la variable x_5 est négatif. Alors, il faut utiliser l'algorithme **dual simplexe** pour la suite:

	x_1	x_2	x_3	x_4 ↓	x_5	b
x_1	1	0	7/3	-1/3	0	7/3
~~x_2~~	0	1	-4/3	1/3	0	8/3
x_5	0	0	-1/3	-2/3	1	-1/3
-Z	0	0	11/3	1/3	0	-83/3

1ère itération →

	x_1	x_2	x_3	x_4	x_5	b
x_1	1	0	5/2	0	-1/2	5/2
x_2	0	1	-3/2	0	1/2	5/2
x_4	0	0	1/2	1	-3/2	1/2
-Z	0	0	-7/2	0	-1/2	-55/2

La solution trouvée est : $x_1 = 5/2$; $x_2 = 5/2$ et $Z = 55/2$; elle sont toujours **non entière**, il faut donc générer une autre coupe. On génère la 2ème coupe pour x_2:

$$x_2 - \frac{3}{2}x_3 + \frac{1}{2}x_5 = \frac{5}{2} \longrightarrow x_2 + \left(\frac{-4}{2} + \frac{1}{2}\right)x_3 + \frac{1}{2}x_5 = \left(\frac{4}{2} + \frac{1}{2}\right) \Leftrightarrow x_2 + \left(-2 + \frac{1}{2}\right)x_3 + \frac{1}{2}x_5 = \left(2 + \frac{1}{2}\right)$$

$$\Leftrightarrow x_2 - 2x_3 - 2 = -\frac{1}{2}x_3 - \frac{1}{2}x_5 + \frac{1}{2} \Rightarrow -\frac{1}{2}x_3 - \frac{1}{2}x_5 + \frac{1}{2} \leq 0 \Rightarrow -\frac{1}{2}x_3 - \frac{1}{2}x_5 \leq -\frac{1}{2}$$

Coupes de Gomory (Gomory Cuts)

Exemple

On va effectuer une itération de l'algorithme dual du simplexe en introduisant la nouvelle coupe:

$$-\frac{1}{2}x_3 - \frac{1}{2}x_5 \leq -\frac{1}{2} \Leftrightarrow -x_3 - x_5 \leq -1 \rightarrow -x_3 - x_5 + x_6 = -1$$

	x_1	x_2	x_3	x_4	x_5 ↓	x_6	b
x_1	1	0	5/2	0	-1/2	0	5/2
x_2 ←	0	1	-3/2	0	1/2	0	5/2
x_4	0	0	1/2	1	-3/2	0	1/2
x_6	0	0	-1	0	-1	1	-1
-Z	0	0	7/2	0	1/2	0	-55/2

→

	x_1	x_2	x_3	x_4	x_5	x_6	b
x_1	1	0	3	0	0	-1/2	3
x_2	0	1	-2	0	0	1/2	2
x_4	0	0	2	1	0	-3/2	2
x_5	0	0	1	0	1	-1	1
-Z	0	0	-3	0	0	-1/2	-27

Toutes les variables sont entières, la solution optimale est : $x_1 = 3$; $x_2 = 2$ et $Z = 27$.

Coupes de Gomory (Gomory Cuts)

Exemple

Pour représenter graphiquement ces 02 coupes, on doit réécrire leurs équations en fonction de x_1 et x_2:

$$(P)\begin{cases} Z_{max} = 5x_1 + 6x_2 \\ x_1 + x_2 + x_3 = 5 \\ 4x_1 + 7x_2 + x_4 = 28 \\ x_1, x_2 \geq 0 \end{cases}$$

- 1ère coupe:

$$-\frac{1}{3}x_3 - \frac{2}{3}x_4 \leq -\frac{1}{3} \Leftrightarrow -x_3 - 2x_4 \leq -1$$

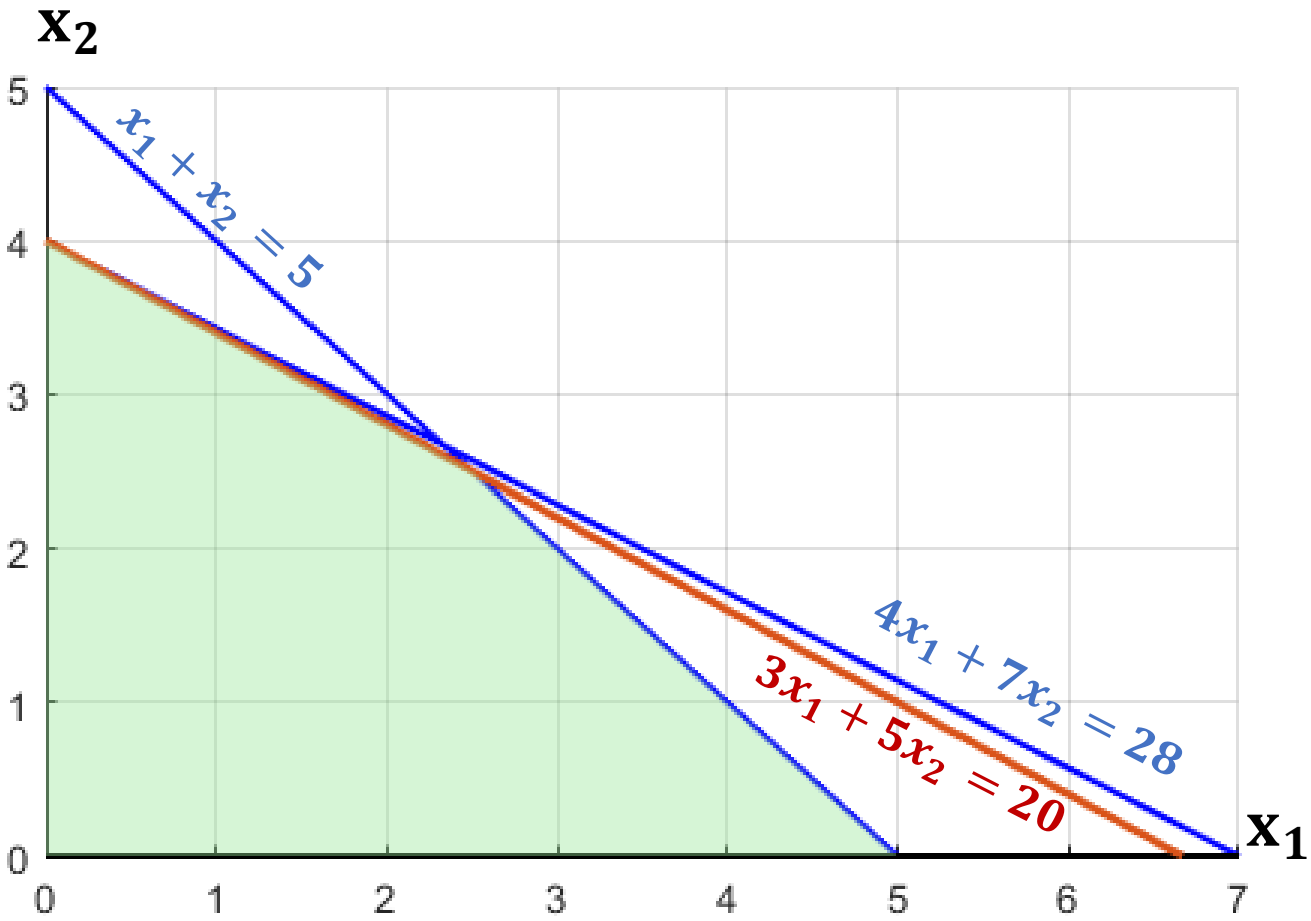
Nous avons:

1. $x_1 + x_2 + x_3 = 5 \Rightarrow x_3 = 5 - x_1 - x_2$
2. $4x_1 + 7x_2 + x_4 = 28 \Rightarrow x_4 = 28 - 4x_1 - 7x_2$

Donc la 1ère coupe peut être représenté par:

$-(5 - x_1 - x_2) - 2(28 - 4x_1 - 7x_2) \leq -1$

$\leftrightarrow 9x_1 + 15x_2 \leq 60 \leftrightarrow 3x_1 + 5x_2 \leq 20$



Coupes de Gomory (Gomory Cuts)

Exemple

Pour représenter graphiquement ces 02 coupes, on doit réécrire leurs équations en fonction de x_1 et x_2:

$$(P)\begin{cases} Z_{max} = 5x_1 + 6x_2 \\ x_1 + x_2 + x_3 = 5 \\ 4x_1 + 7x_2 + x_4 = 28 \\ x_1, x_2 \geq 0 \end{cases}$$

- 2ème coupe:

$$-\frac{1}{2}x_3 - \frac{1}{2}x_5 \leq -\frac{1}{2} \Leftrightarrow -x_3 - x_5 \leq -1$$

Tableau optimal issu de l'introduction de la 1ère coupe :

	x_1	x_2	x_3	x_4	x_5	b
x_1	1	0	5/2	0	-1/2	5/2
x_2	0	1	-3/2	0	1/2	5/2
x_4	0	0	1/2	1	-3/2	1/2
-Z	0	0	-7/2	0	-1/2	-55/2

Nous avons:

1. $x_1 + x_2 + x_3 = 5 \Rightarrow x_3 = 5 - x_1 - x_2$
2. $4x_1 + 7x_2 + x_4 = 28 \Rightarrow x_4 = 28 - 4x_1 - 7x_2$
3. À partir du tableau optimal résultant de l'introduction de la 1ère coupe (à partir duquel nous avons trouver la 2ème coupe), on a:

$$\frac{1}{2}x_3 + x_4 - \frac{3}{2}x_5 = \frac{1}{2} \Leftrightarrow x_3 + 2x_4 - 3x_5 = 1$$

$$\Rightarrow x_5 = \frac{1}{3}(x_3 + 2x_4 - 1) = \frac{1}{3}(5 - x_1 - x_2 + 2(28 - 4x_1 - 7x_2)$$

Donc la 2ème coupe peut être représenté par:

$$-(5 - x_1 - x_2) - (20 - 3x_1 - 5x_2) \leq -1$$

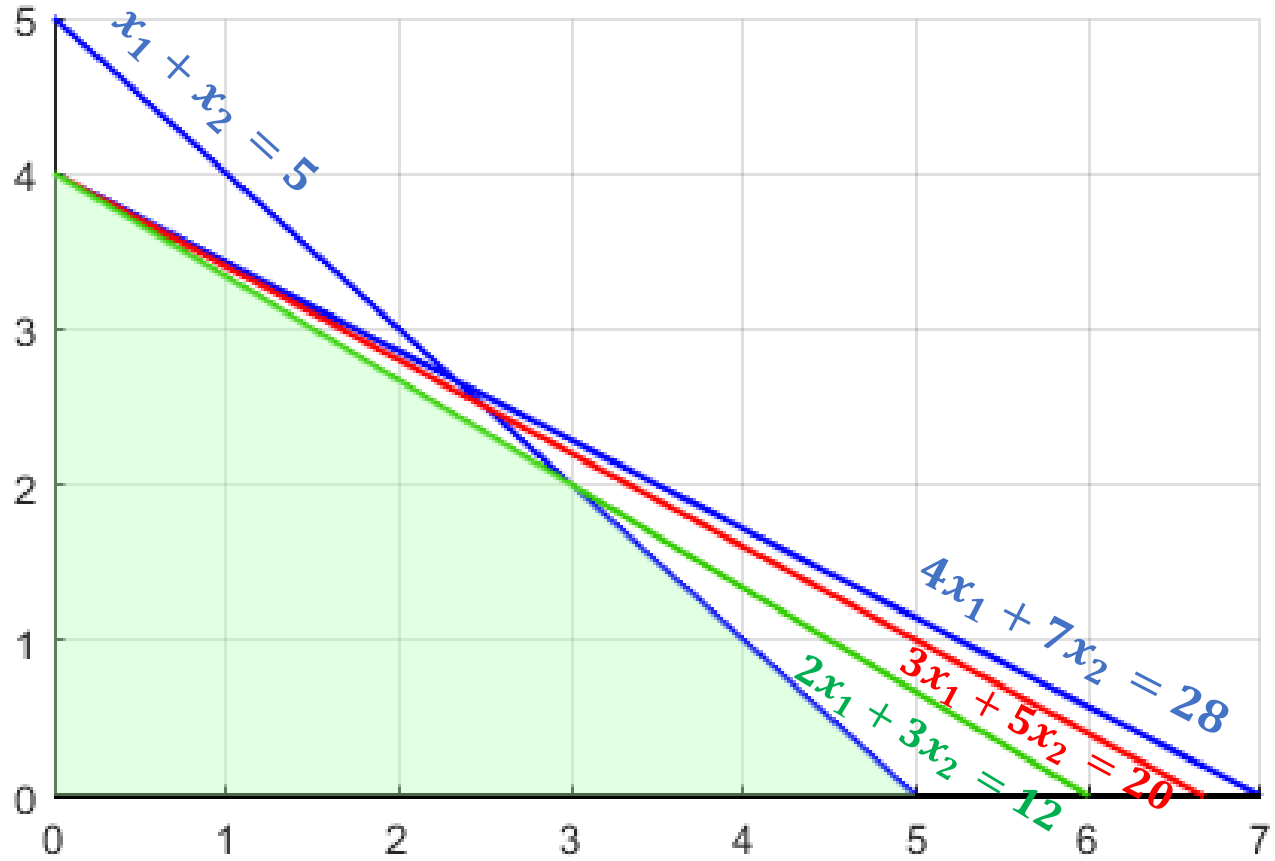
$$\Leftrightarrow 4x_1 + 6x_2 \leq 24 \Leftrightarrow 2x_1 + 3x_2 \leq 12$$

Coupes de Gomory (Gomory Cuts)

Exemple

Pour représenter graphiquement ces 02 coupes, on doit réécrire leurs équations en fonction de x_1 et x_2:

$$(P)\begin{cases} Z_{max} = 5x_1 + 6x_2 \\ x_1 + x_2 + x_3 = 5 \\ 4x_1 + 7x_2 + x_4 = 28 \\ 3x_1 + 5x_2 \leq 20 \\ 2x_1 + 3x_2 \leq 12 \\ x_1, x_2 \geq 0 \end{cases}$$



- 2ème coupe:

$$-x_3 - x_5 \leq -1$$
$$\Leftrightarrow 2x_1 + 3x_2 \leq 12$$

Coupes de Gomory (Gomory Cuts)

Remarques

Les coupes fractionnaires fonctionnent avec l'hypothèse que toutes les variables **sont entières**. Par conséquent, elles n'autorisent pas les valeurs fractionnaires pour les variables d'écart. Exemple :

$\frac{1}{2}x_1 + \frac{1}{6}x_2 + x_3 = \frac{17}{2}$; pour $x_1 = 1$ et $x_2 = 3$; on trouve: $x_3 = \frac{15}{2}$ ⇒ Coupe non faisable

SOLUTIONS:

Multiplier les deux côtés de l'équation de coupe par une valeur appropriée pour que tous les coefficients soient entiers :

$$3x_1 + x_2 + 6x_3 = 51$$

Utiliser des coupes mixtes, qui restreignent l'hypothèse entière à un sous-ensemble de valeurs, par exemple x_1 et x_2 (Hamdy A. Taha, 1975, 198–202)

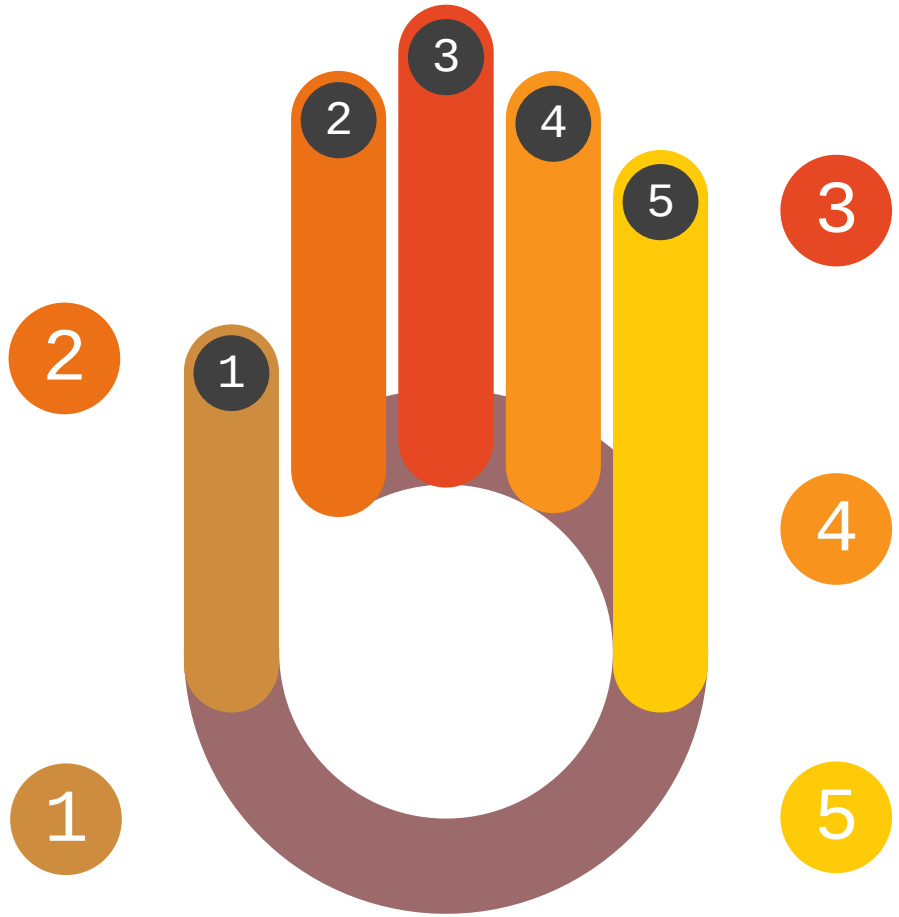
Méthode de Branch and Cut

Algorithme de branch and Cut

Idée Générale

Si la solution optimale obtenue contient des valeurs **non entières**, on applique la méthode **des plans de coupe** (cutting planes). Ces coupes sont ajoutées au programme linéaire pour rapprocher la solution de l'intégralité.

La méthode commence par résoudre le programme linéaire relaxé (sans contrainte d'intégralité) à l'aide de l'algorithme du simplexe



Ensuite, la partie **Branch and Bound** est lancée.

- Le problème est divisé en sous-problèmes (branches) résolus par le simplexe.
- Les solutions fractionnaires servent de bornes supérieures.
- Les solutions entières servent de bornes inférieures..

Un **nœud est élagué** si sa borne supérieure est inférieure à une borne inférieure existante.

Pendant le **Branch & Bound**, de nouvelles coupes peuvent encore être ajoutées :
Coupes globales : valides pour toutes les solutions entières.
Coupes locales : valides seulement pour la sous-branche considérée.

Algorithme de branch and Cut

Pseudocode

```
1.  Ajouter le problème initial (ILP) à la liste L des problèmes actifs
2.  Initialiser :
        x* = null       // meilleure solution entière
        v* = -∞         // valeur de l'objectif correspondante
3.  Tant que L n'est pas vide :
        a. Sélectionner et retirer un problème de L
        b. Résoudre la relaxation linéaire (LP) du problème
        c. Si la solution est infaisable : continuer avec le problème suivant
        d. Soit x la solution obtenue et v sa valeur de l'objectif
           Si v ≤ v* :  continuer avec le problème suivant
        e. Si x est une solution entière :
               v* ← v
               x* ← x
               continuer avec le problème suivant
           Sinon :
               (f). Rechercher des plans de coupe violés par x
                   Si des coupes sont trouvées : les ajouter à la relaxation LP revenir à l'étape 3b
                   Sinon : effectuer un branchement :
                           créer de nouveaux sous-problèmes avec des régions faisables restreintes
                           ajouter ces sous-problèmes à L
4.  Retourner x* comme meilleure solution entière trouvée
```

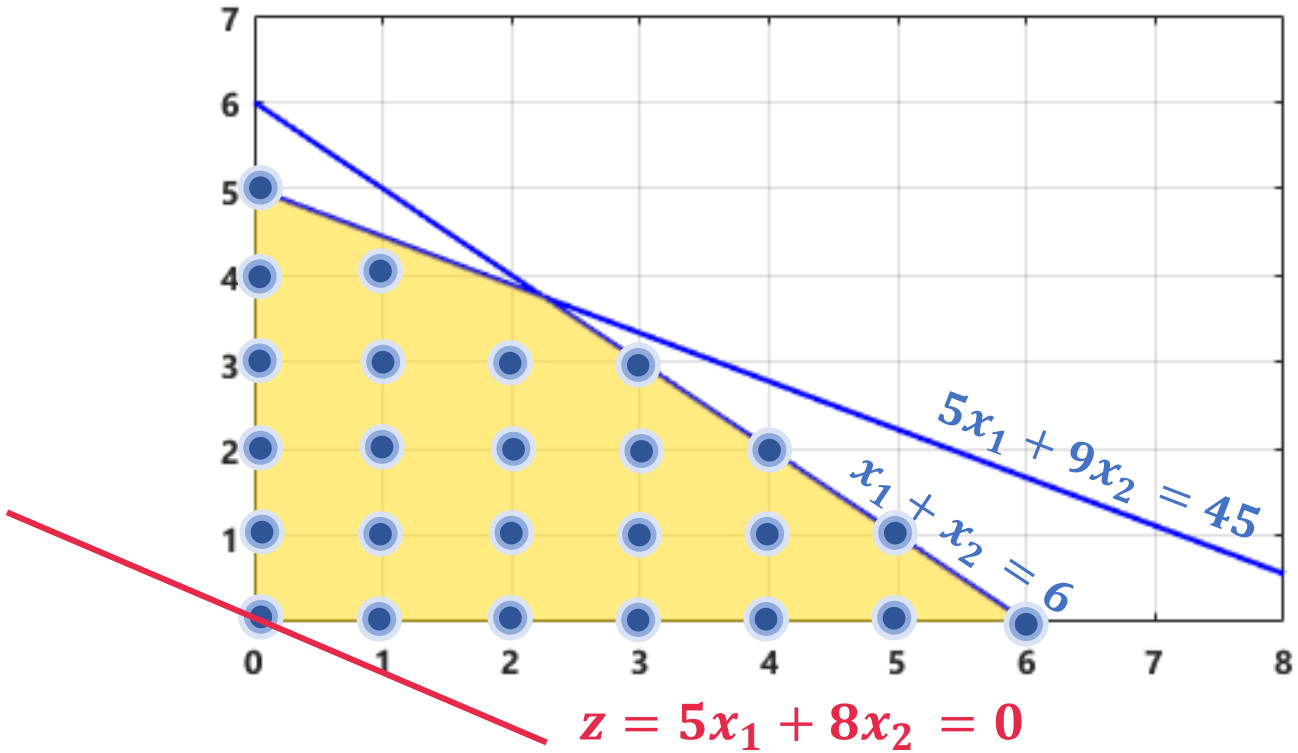
Algorithme de branch and Cut | Exemple

Reprenant le PLNE résolut par la méthode de Branch and Bound :

$$(P)\begin{cases} Z_{max} = 5x_1 + 8x_2 \\ x_1 + x_2 \leq 6 \\ 5x_1 + 9x_2 \leq 45 \\ x_1, x_2 \geq 0, \text{entiers} \end{cases}$$

Tableau optimal de la relaxation continu:

	x_1	x_2	x_3	x_4	b
x_1	0	1	9/4	-1/4	9/4
x_2	1	0	-5/4	1/4	15/4
-Z	0	0	-5/4	-3/4	-165/4

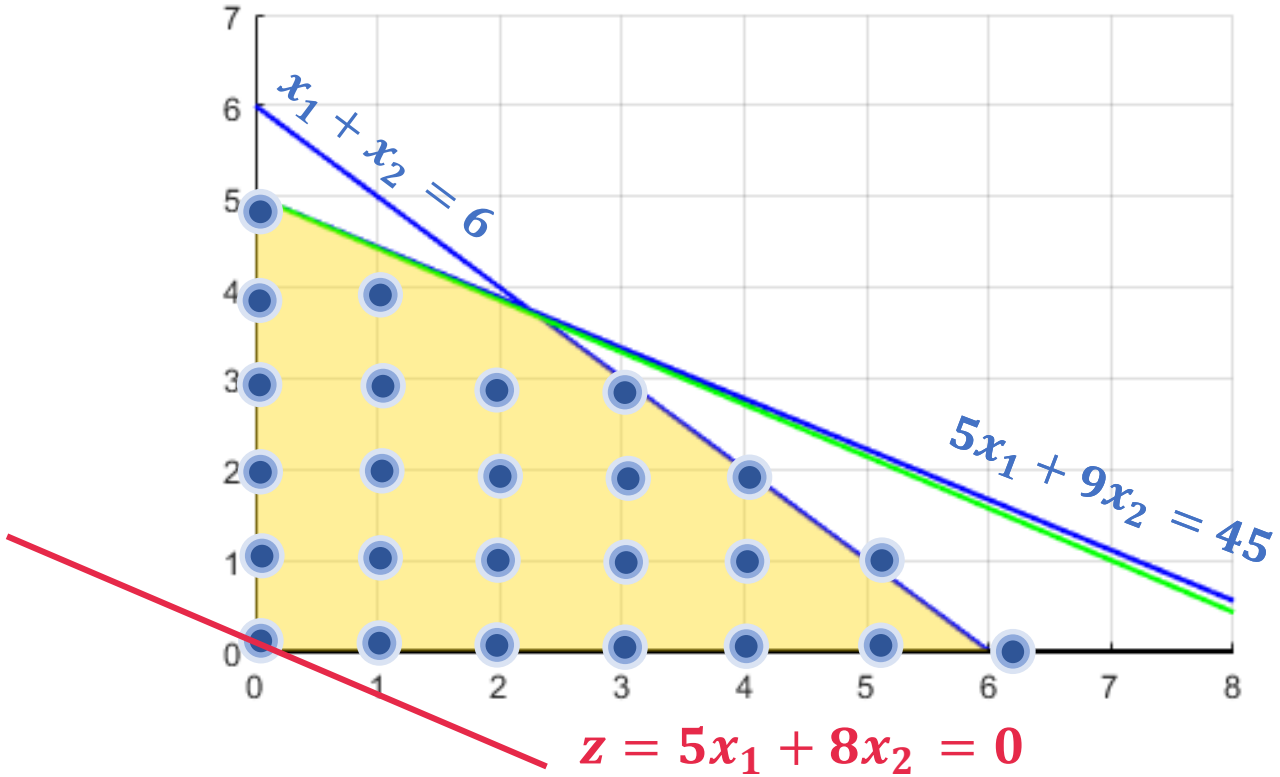


Coupe de gomory pour x_1 → $4x_1 + 7x_2 \leq 35$

Algorithme de branch and Cut | Exemple

On rajoute la coupe de **gomory** et on résout le nouveau relaxé:

$$(\mathrm{P})\begin{cases} Z_{max} = 5x_1 + 8x_2 \\ x_1 + x_2 \leq 6 \\ 5x_1 + 9x_2 \leq 45 \\ \mathbf{4x_1 + 7x_2 \leq 35} \\ x_1, x_2 \geq 0 \end{cases}$$



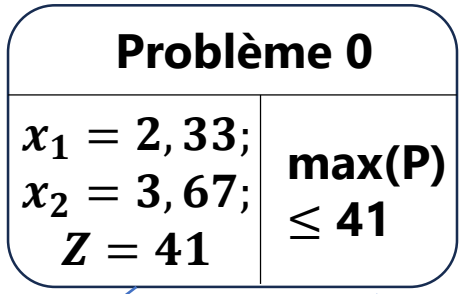
Solution optimale continu:
$x_1 = 2,33;\ x_2 = 3,67; Z = 41$

On effectue un **branch** sur x_2 (ayant la plus grande partie fractionnaire)

Algorithme de branch and Cut

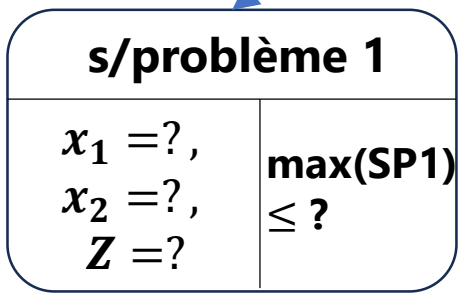
Exemple

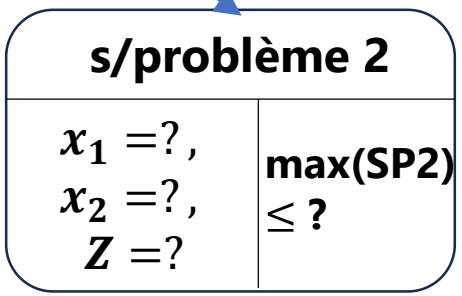
Pour SP1, on procède à la résolution par tableaux du simplexe. Si solution entière, on s'arrête. Sinon, on rajoute une coupe de **gomory** et on refait la résolution.



$x_2 \geq 4$

$x_2 \leq 3$





$$(SP1)\begin{cases} Z_{max} = 5x_1 + 8x_2 \\ x_1 + x_2 \leq 6 \\ 5x_1 + 9x_2 \leq 45 \\ 4x_1 + 7x_2 \leq 35 \\ x_2 \geq 4 \\ x_1, x_2 \geq 0 \end{cases}$$

$$(SP2)\begin{cases} Z_{max} = 5x_1 + 8x_2 \\ x_1 + x_2 \leq 6 \\ 5x_1 + 9x_2 \leq 45 \\ 4x_1 + 7x_2 \leq 35 \\ x_2 \leq 3 \\ x_1, x_2 \geq 0, \end{cases}$$

Algorithme de branch and Cut

Exemple

$$(SP1)\begin{cases} Z_{max} = 5x_1 + 8x_2 \\ x_1 + x_2 \leq 6 \\ 5x_1 + 9x_2 \leq 45 \\ \mathbf{4x_1 + 7x_2 \leq 35} \\ \mathbf{x_2 \geq 4} \\ x_1, x_2 \geq 0 \end{cases}$$

Tableau optimal de la relaxation continu:

	x_1	x_2	x_3	x_4	x_5	x_6	b
x_2	0	1	0	0	0	-1	4
x_3	0	0	1	0	-1/4	-3/4	1/4
x_4	0	0	0	1	-5/4	1/4	1/4
x_1	1	0	0	0	1/4	7/4	7/4
-Z	0	0	0	0	-5/4	-3/4	163/4

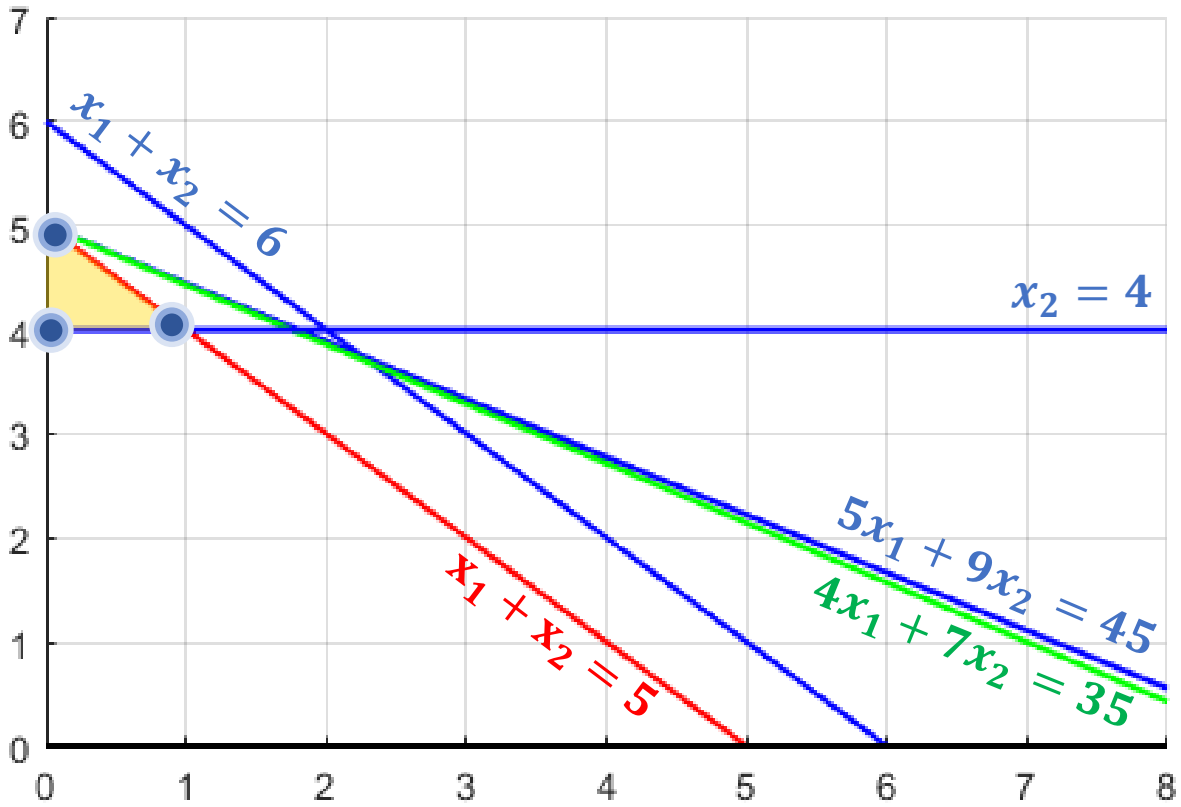
Solution optimale continu: $\boldsymbol{x_1 = 1,75}$; $\boldsymbol{x_2 = 4}$; $\boldsymbol{Z = 40,75}$

On effectue un **cut** de ce PL. la coupe de gomory correspondante à la variable x_1 est :

$\mathbf{x_1 + x_2 \leq 5}$

Algorithme de branch and Cut | Exemple

$$(SP1)\begin{cases} Z_{max} = 5x_1 + 8x_2 \\ x_1 + x_2 \leq 6 \\ 5x_1 + 9x_2 \leq 45 \\ \mathbf{4x_1 + 7x_2 \leq 35} \\ \mathbf{x_2 \geq 4} \\ \mathbf{x_1 + x_2 \leq 5} \\ x_1, x_2 \geq 0 \end{cases}$$



Solution optimale continu:

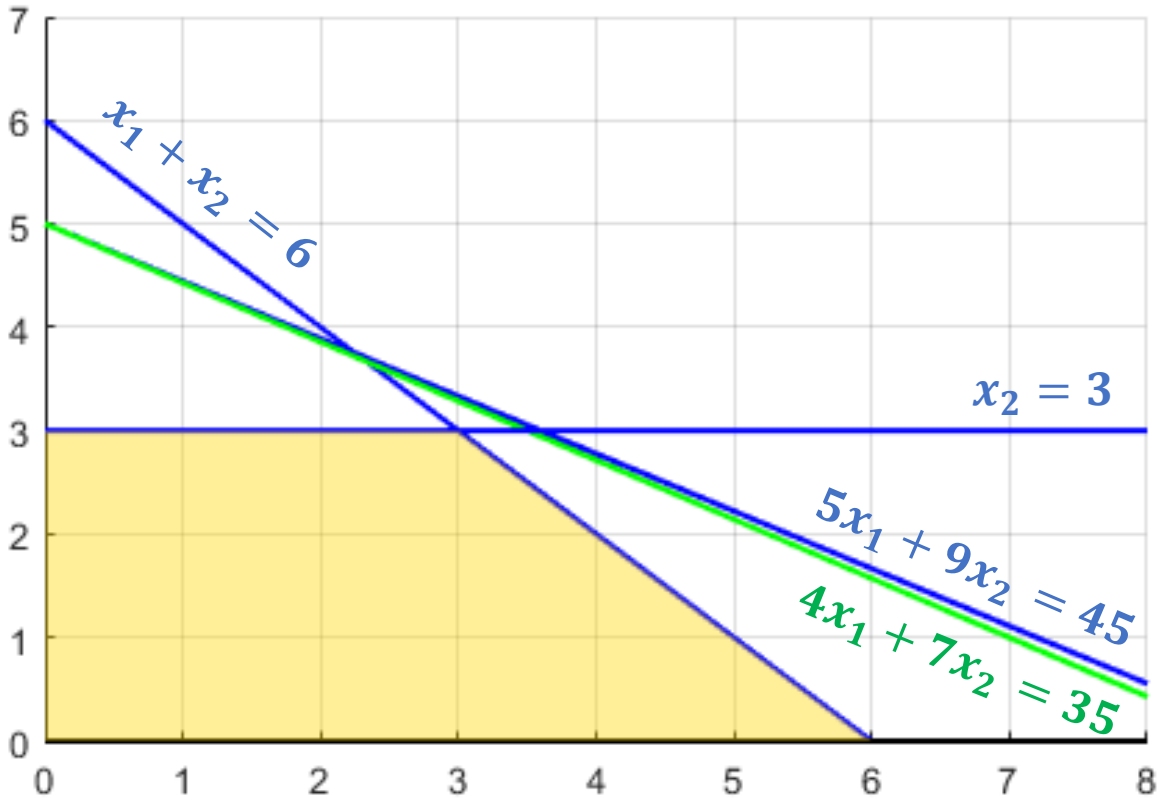
$\boldsymbol{x_1 = 0;\ x_2 = 5; Z = 40}$

La solution est entière donc on s'arrete pour ce nœud et on résout le SP2.

Algorithme de branch and Cut

Exemple

$$(SP2)\begin{cases} Z_{max} = 5x_1 + 8x_2 \\ x_1 + x_2 \leq 6 \\ 5x_1 + 9x_2 \leq 45 \\ \mathbf{4x_1 + 7x_2 \leq 35} \\ \mathbf{x_2 \leq 3} \\ x_1, x_2 \geq 0, \end{cases}$$



Solution optimale continu: $\boldsymbol{x_1 = 3}$; $\boldsymbol{x_2 = 3}$; $\boldsymbol{Z = 39}$

La solution est entière donc on s'arrete pour ce nœud. On élague ce nœud (pas besoin de faire un cut) car solution entière de plus on connait une solution entière meilleure que celle là.

Algorithme de branch and Cut

Exemple

- Le **nœud racine** explore la relaxation.
- Les **nœuds fils** intègrent des contraintes de **branchements**.
- À chaque étape, on **résout**, on **teste l'intégralité**, et on **met à jour les bornes**.
- La méthode s'arrête quand tous les nœuds sont explorés ou élagués.

