Outils de Recherche opérationnelle en Génie MTH 8414

Programmation en nombre entier:

Méthode de résolution et astuces de modélisation

Types de problème d'optimisation

- Programme Linéaire en Nombre Entier (PLNE ou IP) : $X \subseteq \mathbb{Z}^n$
- Programme Linéaire en Nombre Entier Mix (MIP) :

$$\begin{array}{rl} \min\limits_{x \in X} & c^T x \\ \text{s.t.} & Ax \leq b \\ \text{where,} & X \subseteq \mathbb{Z}^{n_i} \times \mathbb{R}^{n_r} \end{array}$$

- Ces problèmes sont théoriquement très difficiles, mais en pratique ils peuvent (souvent) être résolus très rapidement.

Formulation PLNE

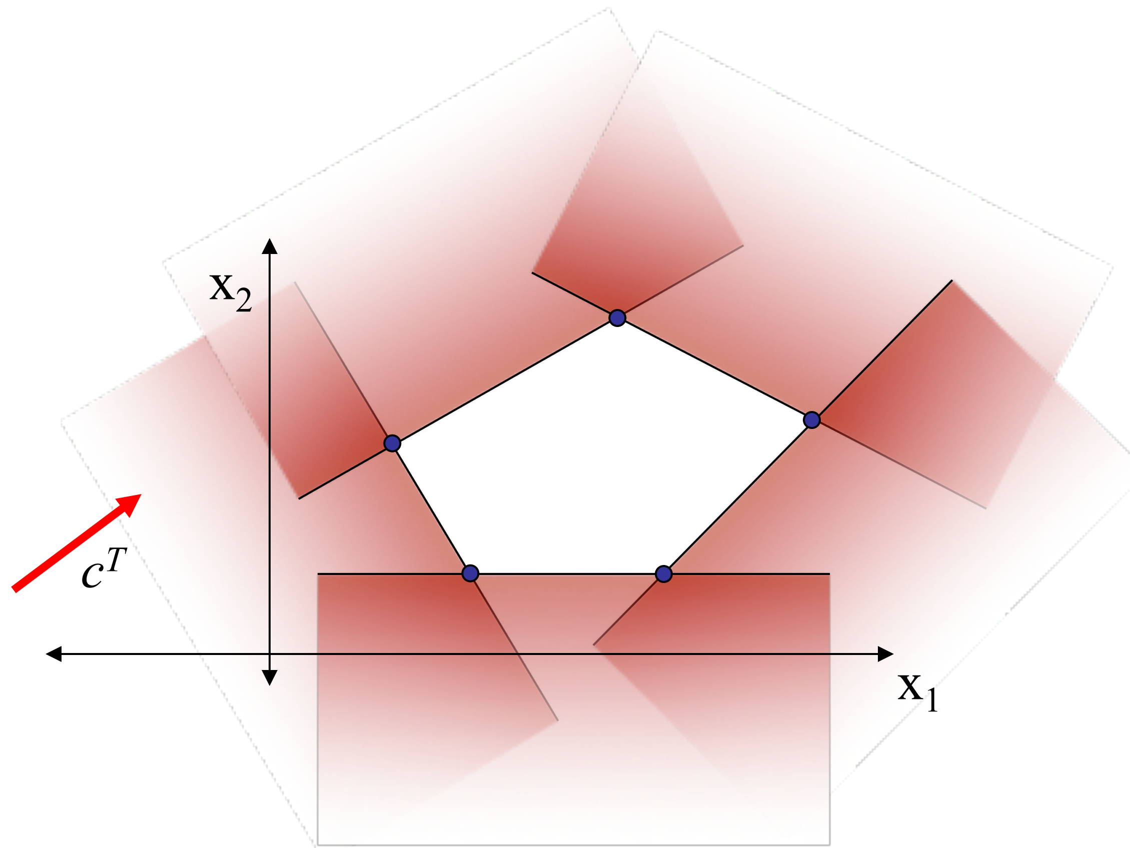
$$
\begin{aligned}
\min \sum_{s \in \Omega} c_s x_s & \\
s.t. \sum_{s \in \Omega} a_{is} x_s \quad & \geq \quad d_i \quad \forall i \in T \\
x_s \quad & \cancel{\geq} \quad 0, \text{entier}, \forall s \in \Omega
\end{aligned}
$$

on obtient un PL nommé
"Relaxation Linéaire"

- Résolu par séparation et évaluation progressive
  - on branche sur les variables de décision
  - la relaxation linéaire nous donne des bornes inférieures.

Méthodes de résolution pour PL

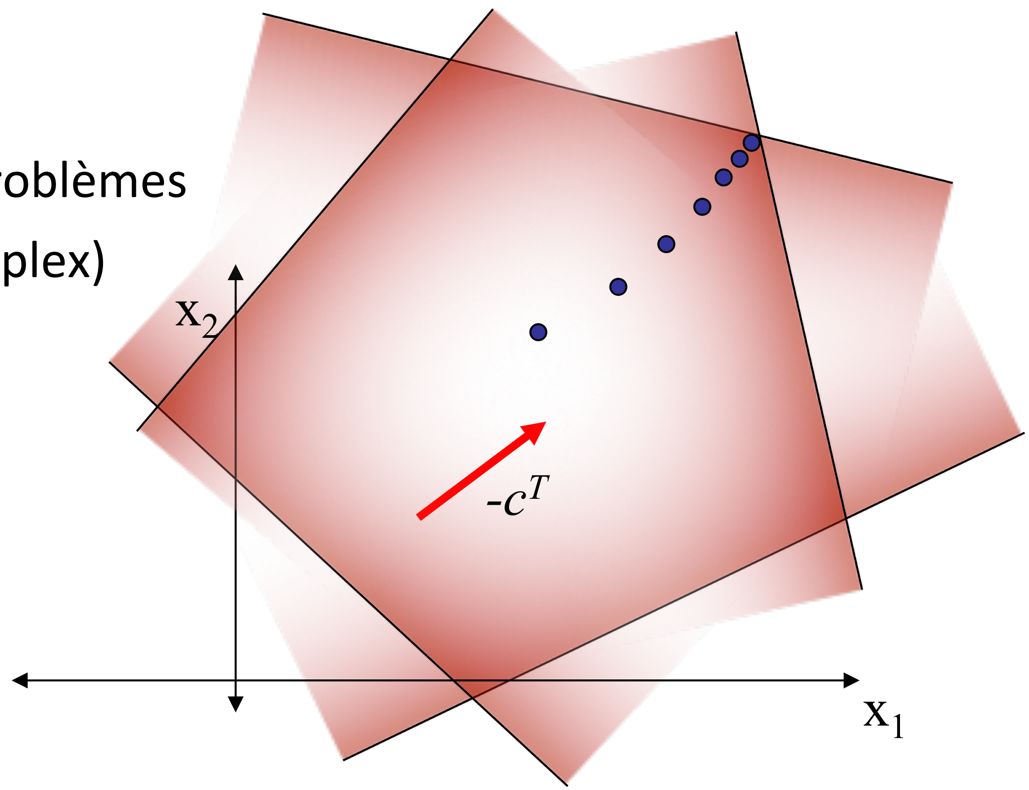
- Algorithme du simplex
  - Une solution optimale se trouve nécessairement sur un point extrême.
  - Donc on peut la trouver en parcourant les arêtes du polyèdre.



Méthodes de résolution pour PL

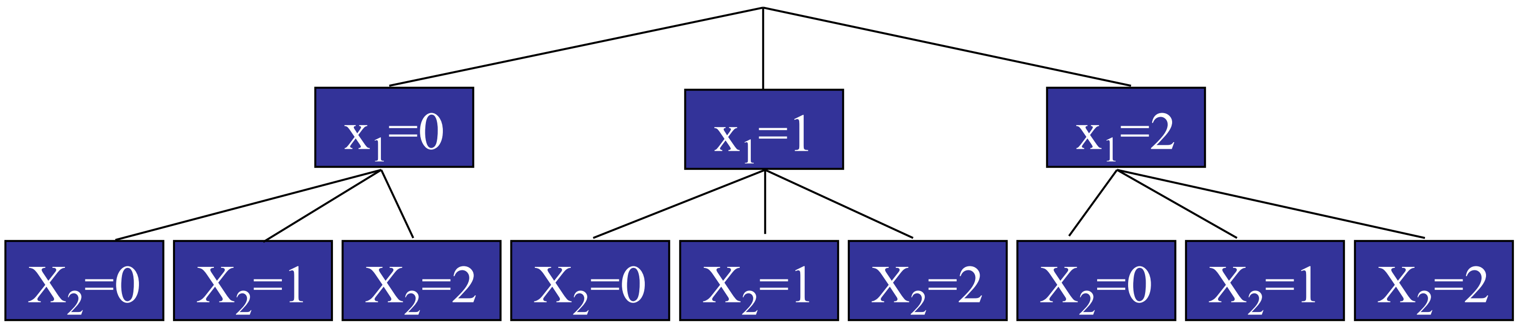
- Méthode par Points Intérieurs
  - Méthode dites “Barrières”
  - Formulation “Primal-Dual”
  - Pas de Newton

- Avantages
  - Permet de résoudre de très gros problèmes
  - Preuve d’optimalité (comme le simplex)



Méthodes de résolution pour PLNE

- Énumération (Recherche Arborescente, Programmation Dynamique)

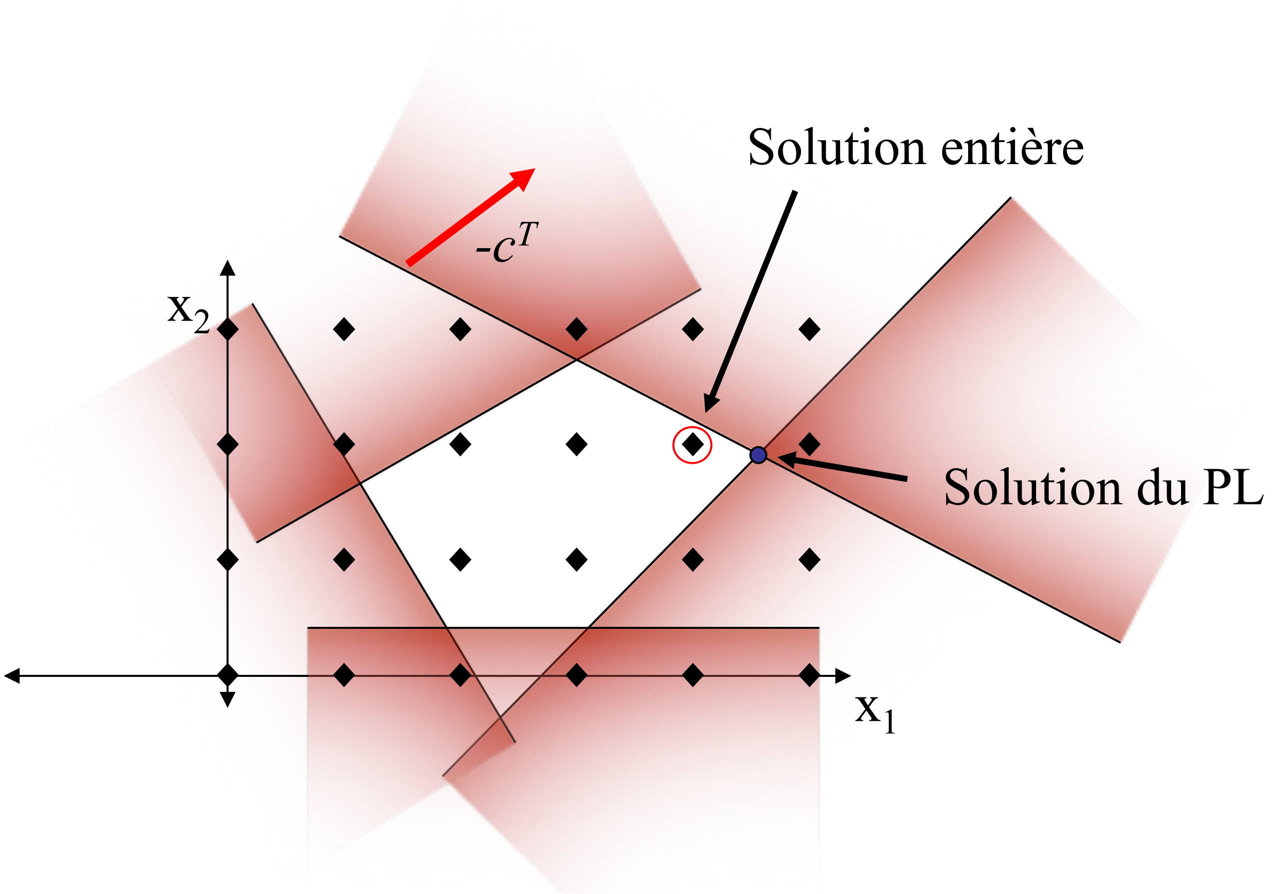


- Garantie de trouver une solution réalisable entière.

- Mais le temps de calcul croît exponentiellement avec la taille.

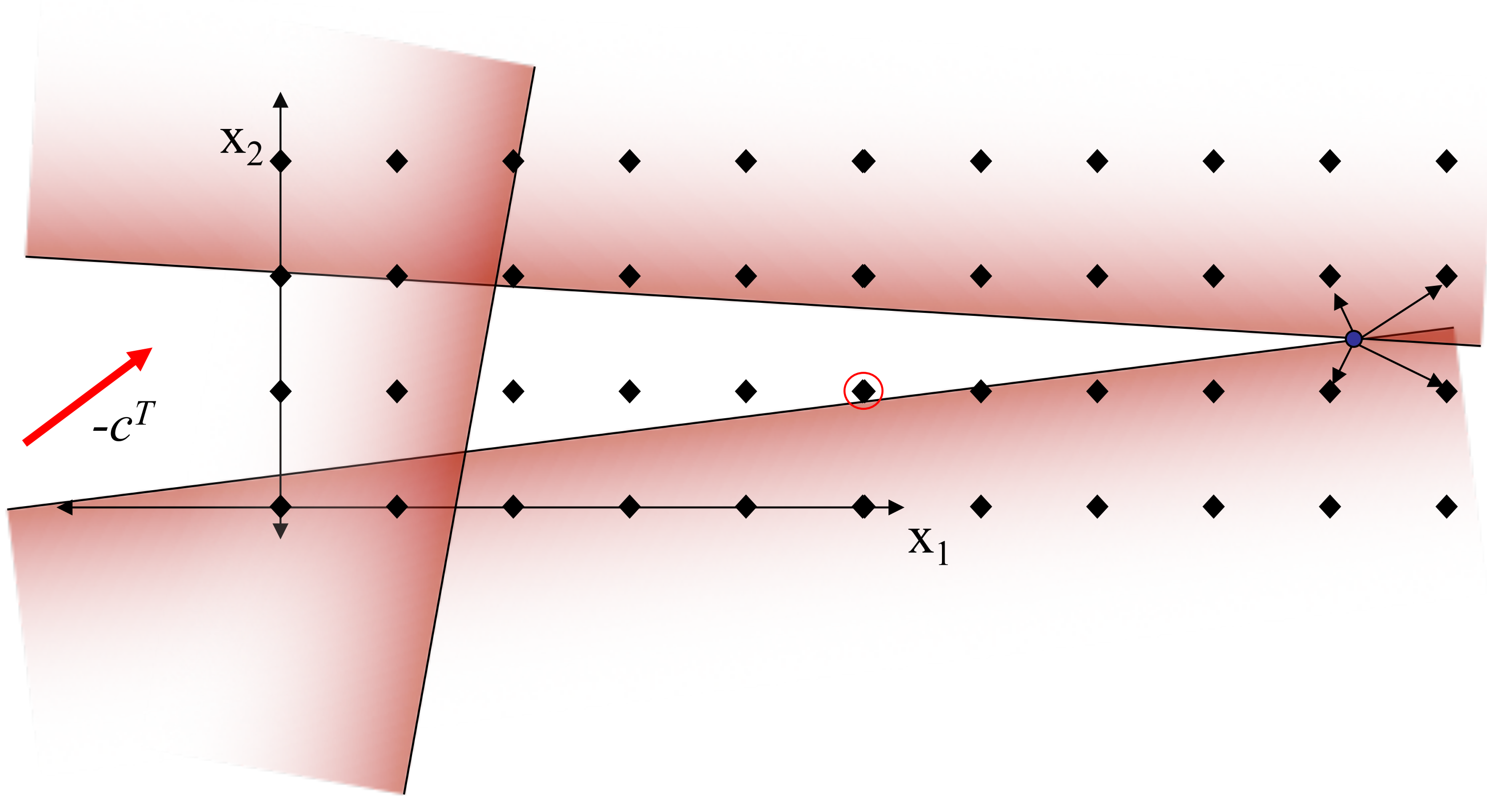
Méthodes de résolution pour PLNE

- Résoudre un PL puis arrondir ?



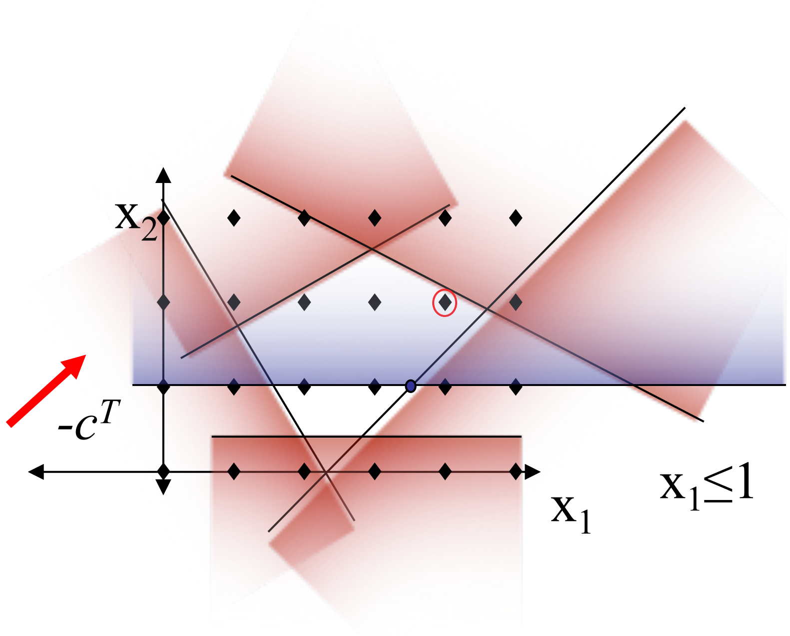
Méthodes de résolution pour PLNE

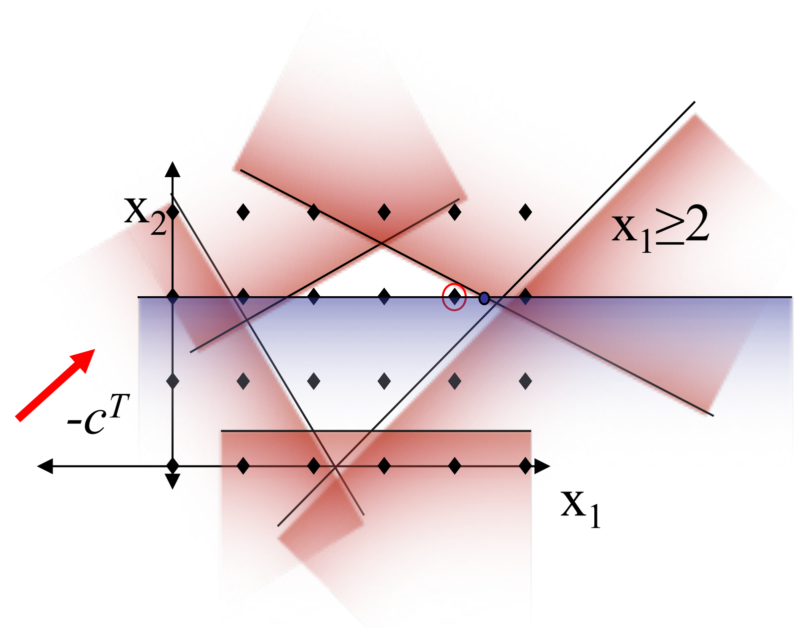
- Le PL fournit une borne inf (ou sup si on maximise) sur la valeur du PLNE.
- Mais en arrondissant, on peut être très loin d'une solution entière…



Approche combinée pour PLNE.

- On peut combiner les deux approches
  - Résoudre le PL pour obtenir une solution.
  - Créer deux sous-problèmes en ajoutant des contraintes.





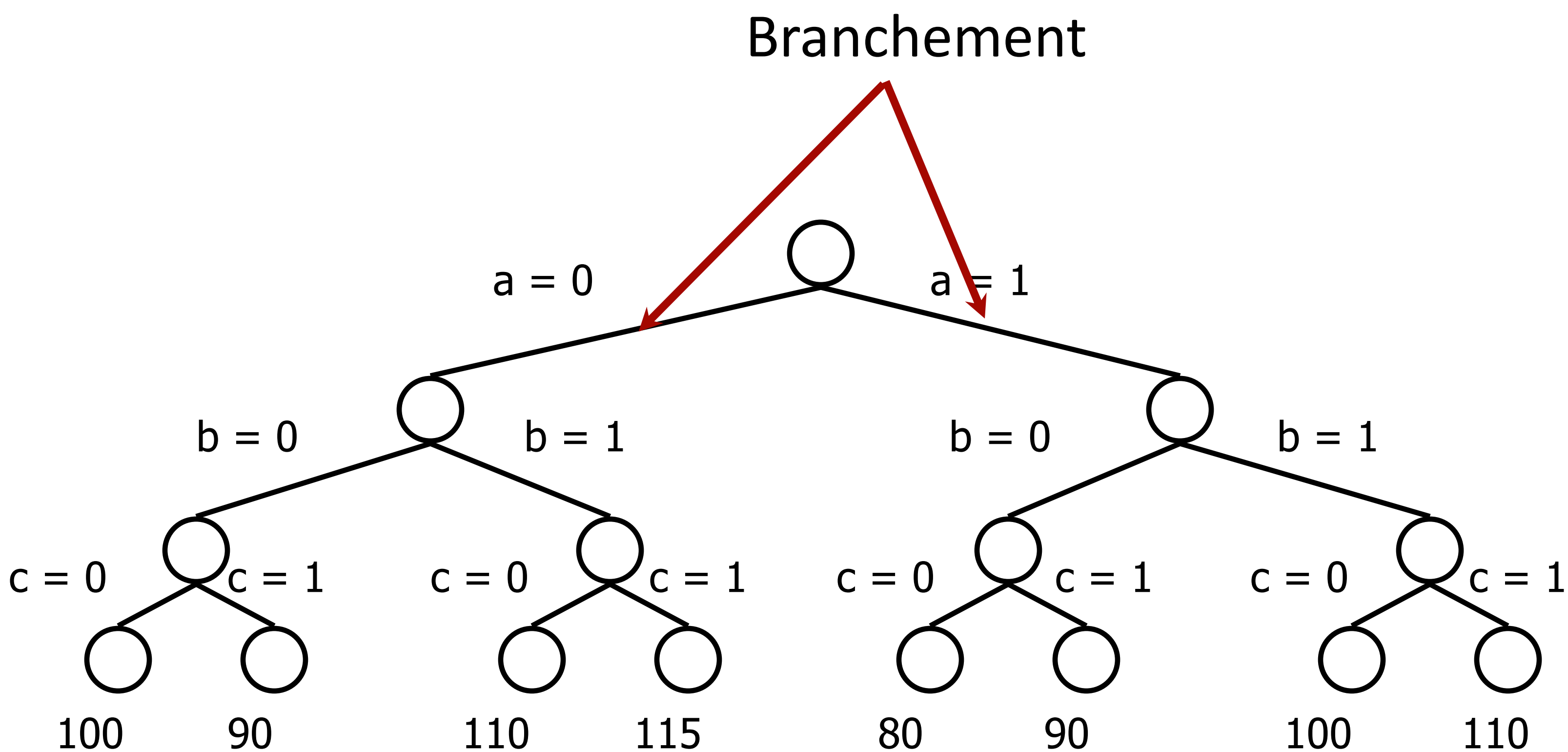
Séparation et évaluation progressive

- Principe
  - Chercher systématiquement toutes les combinaisons variables-valeurs possibles.
  - Utiliser une heuristique pour déterminer sur quelle variable brancher.
  - Utiliser les bornes inférieures pour limiter la recherche.

- Construire un arbre de recherche.

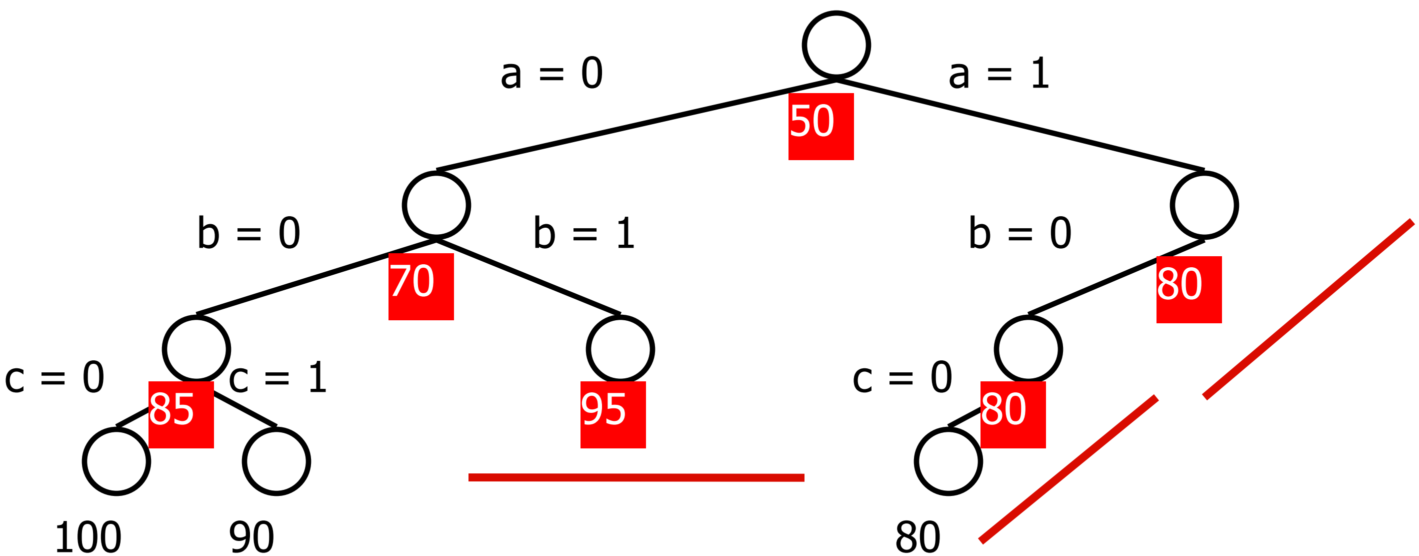
SEP: le branchement

- Imaginez un problème avec 3 variables
  - a, b, c ϵ {0, 1}



SEP: utilisation des bornes inférieures

- Si nous pouvions calculer une borne sur le coût minimal d'un noeud.



Séparation et évaluation progressive

- Mieux connu sous le nom anglais "branch and bound"
- Branch: assigne heuristiquement une valeur à une variable
  - Crée deux sous problèmes
- Bound: comparer la borne inférieure à la meilleure solution connue
  - Ça ne vaut pas la peine d'explorer le sous-arbre si
    - Minimisation: si BorneInf >= MeilleureSolution,
    - Maximisation: si BorneSup <= MeilleureSolution,

SÉP pour résoudre des PLNE

- Généralement la borne inférieure = la relaxation linéaire.
  - On l'obtient en « relaxant » les contraintes d'intégrité.
- On choisit une variable non entière et on la force soit à :
  - être plus grande ou égale à l'entier supérieur ou
  - être plus petite ou égale à l'entier inférieur.

Branch and Bound: un exemple

$$
\begin{aligned}
&\max z = 10x_1 + 50x_2 \\
&\text{s.c.} \\
&-x_1 + 2x_2 \leq 5 \\
&\quad x_1 + 2x_2 \leq 14 \\
&\quad x_1 \qquad\quad \leq 8 \\
&\quad x_1,\ x_2 \quad \geq 0 \\
&\quad x_1,\ x_2 \ \text{entiers}
\end{aligned}
$$

← Relaxation linéaire (ou continue)

Branch and Bound: un exemple

Premier noeud

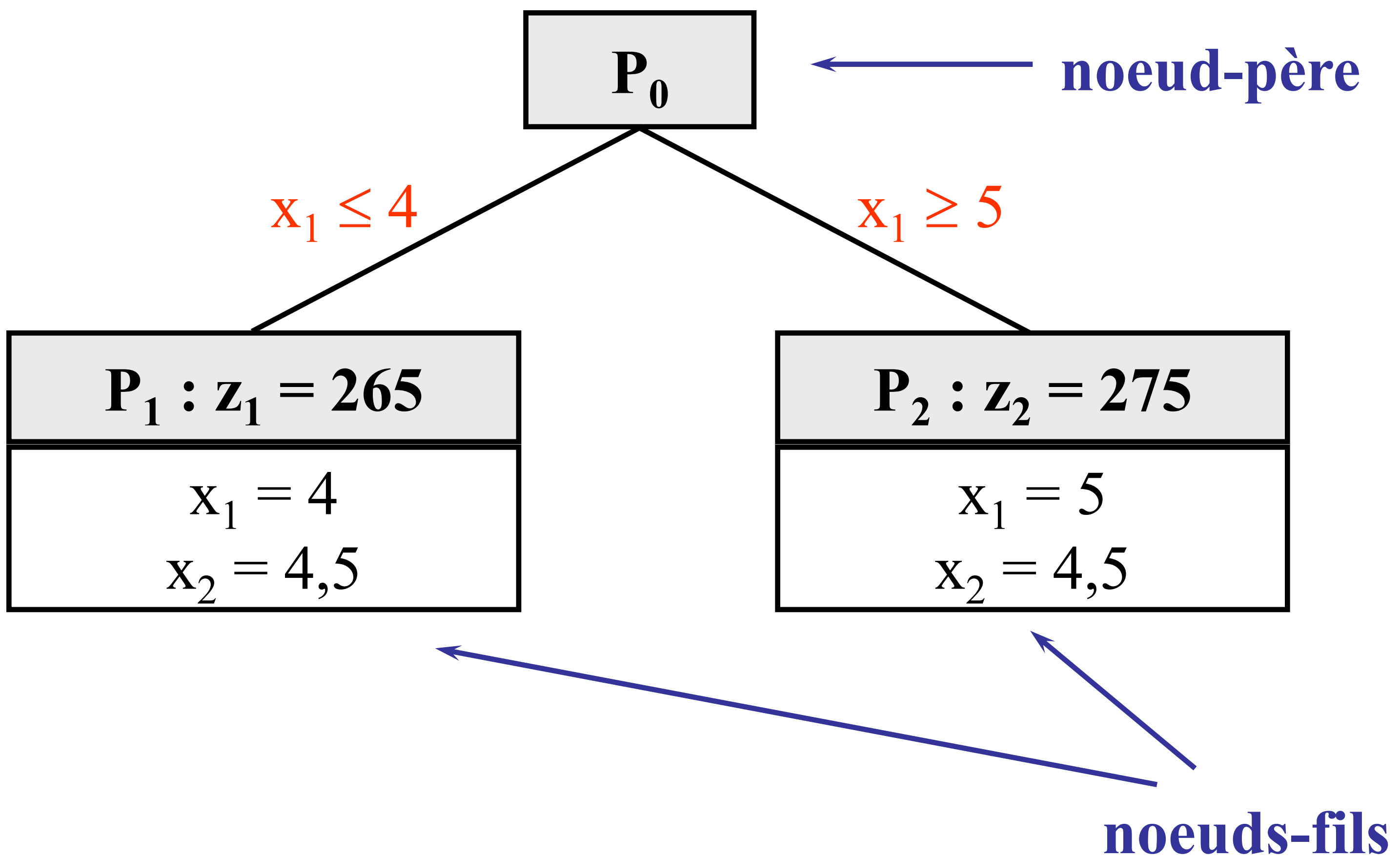
(solution optimale de la relaxation linéaire)

$P_0 : z_0 = 282{,}5$
$x_1 = 4{,}5$ $x_2 = 4{,}75$

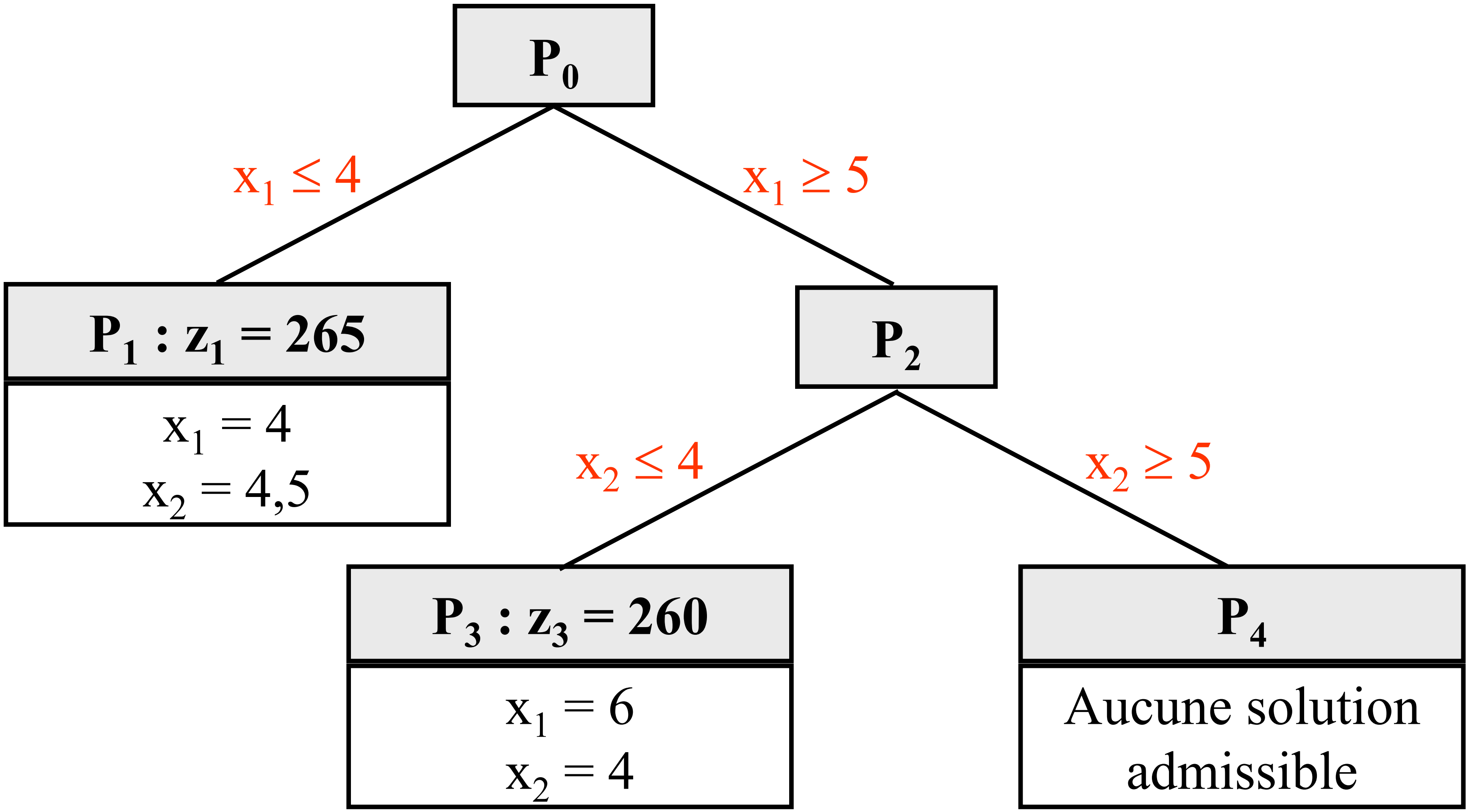
← **Valeur optimale**

← **Variables non nulles**

Branch and Bound

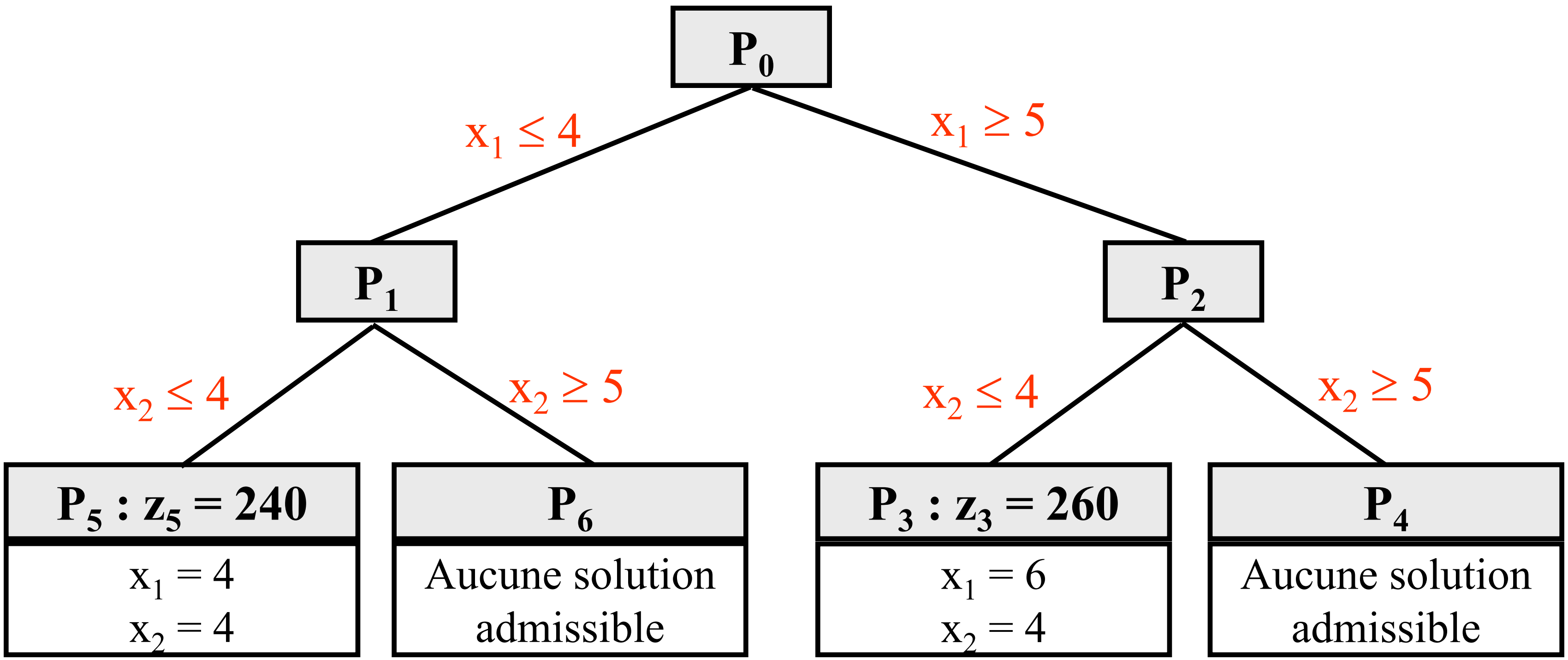


Branch and Bound: un exemple



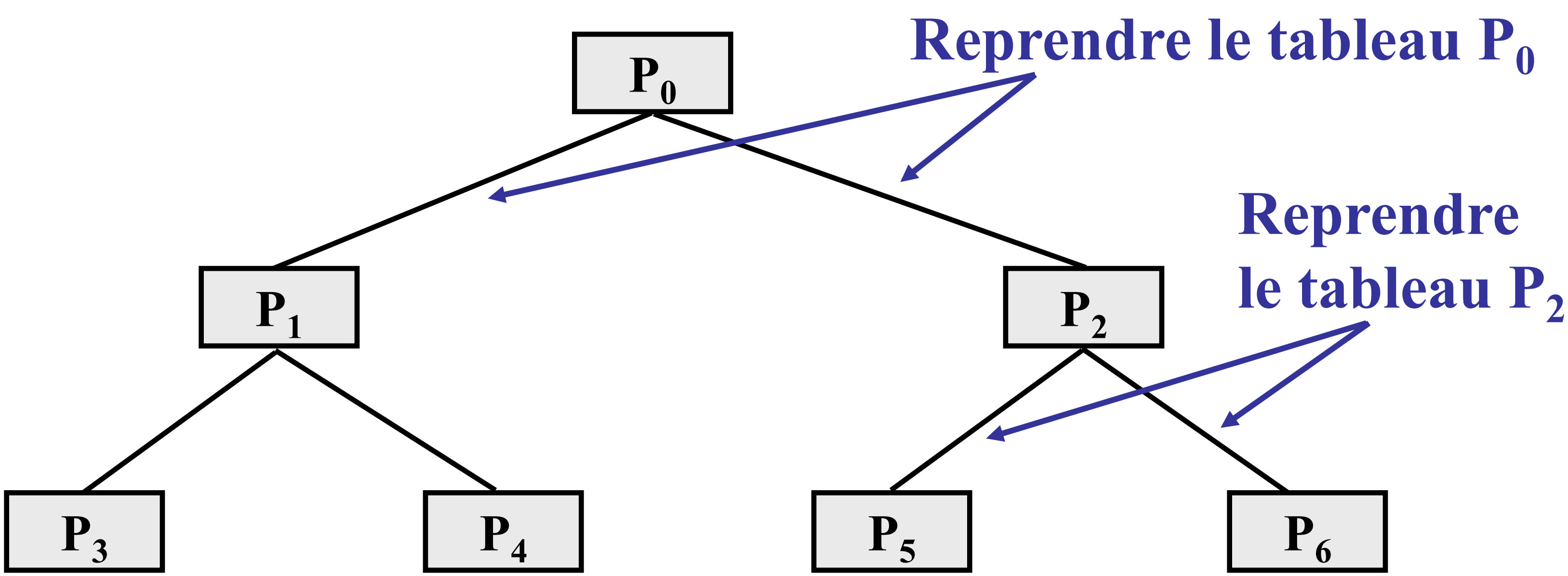
Branch and Bound

- Au final…



Calculs incrémentaux

- En règle générale, pour calculer une solution optimale d'un nœud-fils, il sera plus rapide de modifier le tableau optimal du nœud père plutôt que de reprendre les calculs de l'algorithme du simplexe à partir de leur début.



Algorithme (problème de minimisation)

Les notations suivantes sont utilisées :

L : ensemble des sous-problèmes actifs;

z_U : la borne supérieure sur la valeur optimale de *MIP* ;

z^i_{LP} : la valeur optimale du problème linéaire *i* ;

$\underline{z}^j_{LP}$: la borne inférieure sur la valeur optimale du sous-problème *j* ;

X^* : La meilleure solution réalisable.

L'algorithme comprend 6 étapes :

– **Étape 1 : Initialisation**

L = {relaxation initiale}, $z_U = \infty$.

– **Étape 2 : Test d'optimalité**

Si $L = \varnothing$, x^* est la solution optimale.

– **Étape 3**

Choisir un sous-problème i et l'éliminer de la liste L.

– **Étape 4**

Résoudre la relaxation linéaire de i. Si elle n'est pas réalisable, allez à l'étape 3
Sinon, poser z^i_{LP} et x^i la valeur et la solution optimales obtenues.

– **Étape 5**

Si $z^i_{LP} \geq z_U$, aller à l'étape 2. Si x^i n'est pas entière, aller à l'étape 6.

Sinon $z_U = z^i_{LP}$, $x^* = x^i$.

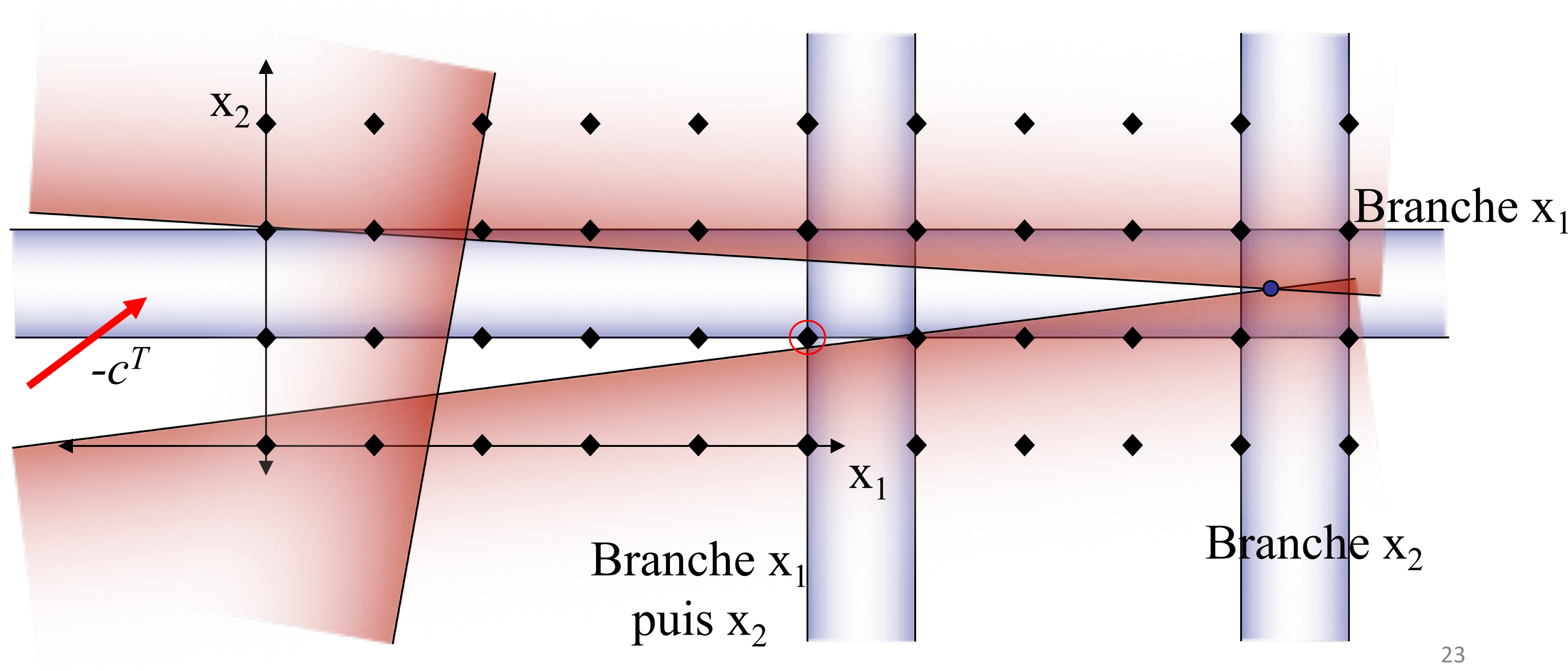
Éliminer de L tous les sous-problèmes j tels que $z^j_{LP} \geq z_U$ et aller à l'étape 2.

– **Étape 6**

Choisir une variable binaire ayant une valeur fractionnaire dans la solution x^i et subdiviser le problème *i* à partir de cette variable. Ajouter les nouveaux problèmes à L.

Algorithme (remarque)

- Pour que l'algorithme soit complètement défini, on doit fixer:
  – à l'étape 3, la sélection du sous-problème à résoudre et
  – à l'étape 6, la règle de séparation du nœud courant.
- Ces deux règles (choix de nœuds et choix de variables) sont cruciales quant à l'efficacité de l'approche de séparation et d'évaluation progressive.



Un autre exemple

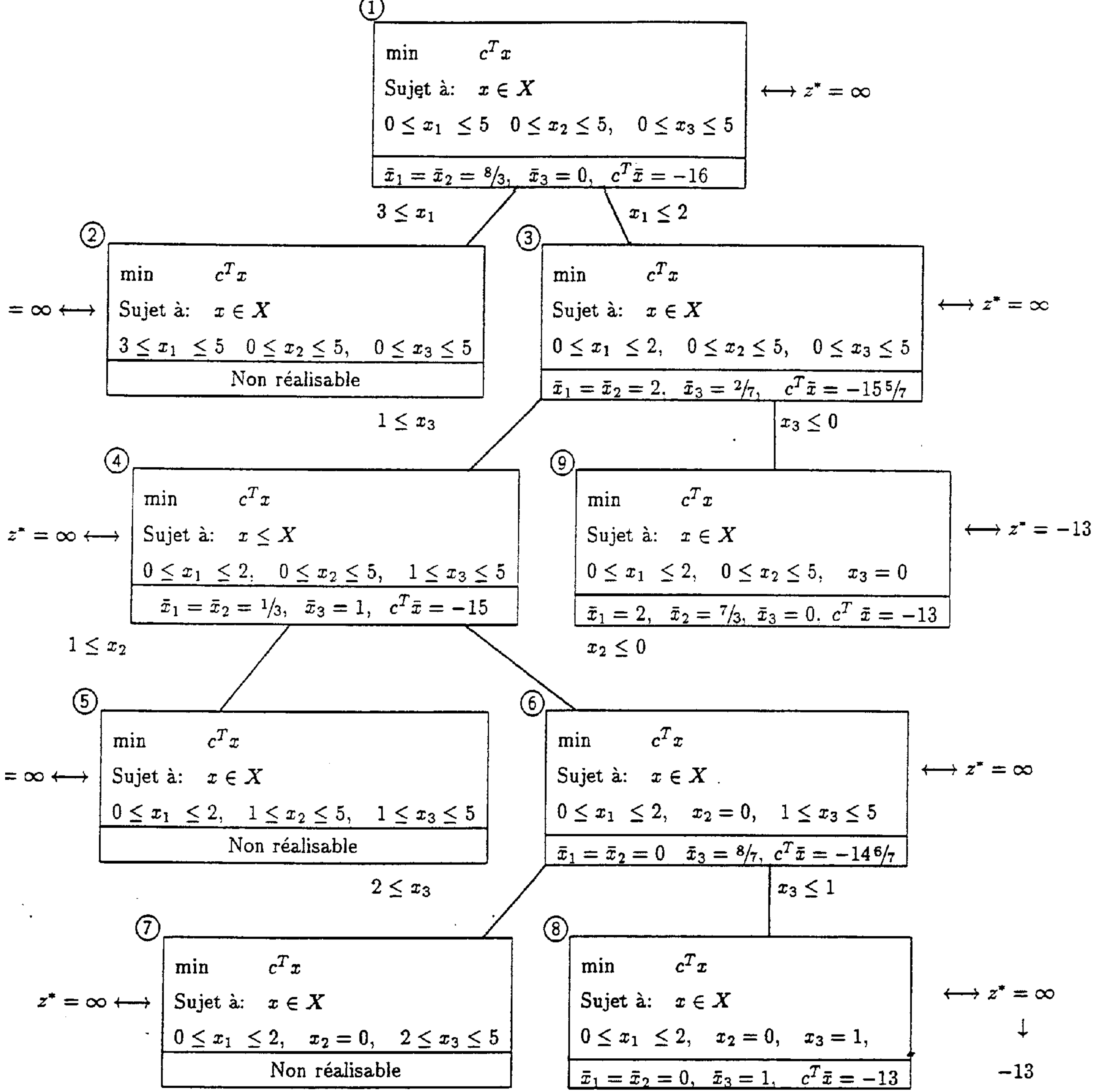
- Soit le problème de PLNE suivant:

$$
\begin{aligned}
(P):\ \ \min \quad & z = -3x_1 - 3x_2 - 13x_3 \\
\text{sujet à :} \quad & \\
& -3x_1 + 6x_2 + 7x_3 \leq 8 \\
& 6x_1 - 3x_2 + 7x_3 \leq 8 \\
& x_1 \leq 5 \\
& x_2 \leq 5 \\
& x_3 \leq 5 \\
& x_1, x_2, x_3 \geq 0 \text{ et entières.}
\end{aligned}
$$

- Et la notation

$$X = \{x \in I^3 : -3x_1 + 6x_2 + 7x_3 \leq 8,\ 6x_1 - 3x_2 + 7x_3 \leq 8\}$$

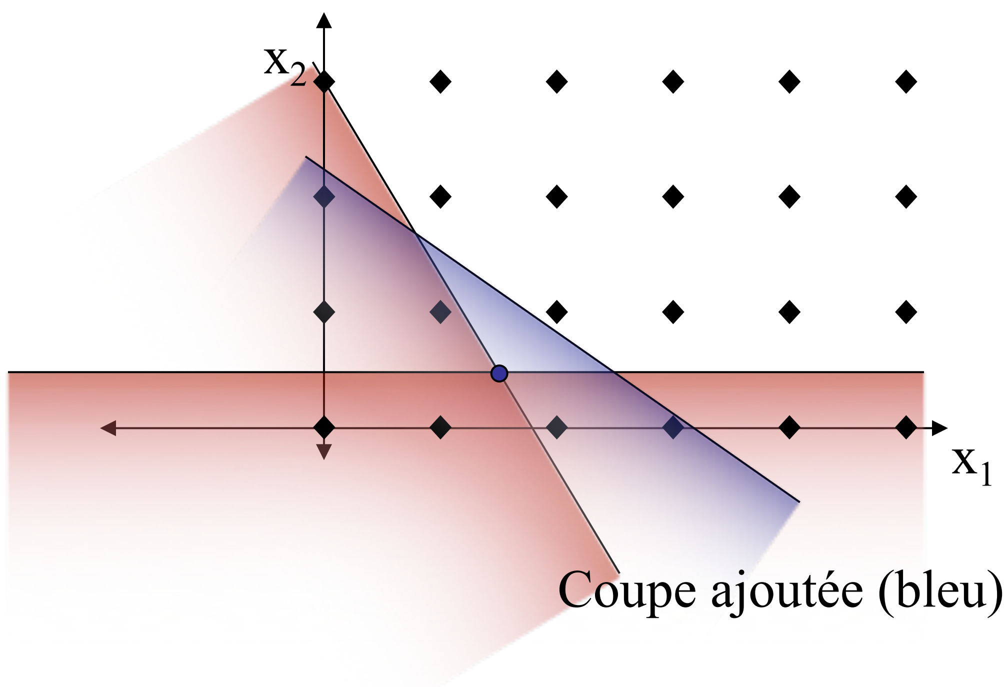
$$z = c^T x = -3x_1 - 3x_2 - 13x_3$$



Solution optimale $x_1 = x_2 = 0, \quad x_3 = 1, \quad c^T x = -13$

Ajout de plans coupant (branch and cut)

- L'idée est d'ajouter des coupes au PL pour améliorer la qualité de la borne.



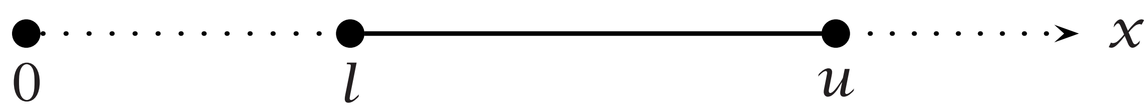
- Toutes les solutions entières sont préservées
- La solution actuelle du PL devient non réalisable.

Trucs et astuces de modélisation

- Comment modéliser les cas où l'on est en présence de:
  - variables ont des domaines discontinus;
  - certaines ressources qui ont des coûts fixes;
  - disjonctions de contraintes;
  - contraintes conditionnelles
  - de SOS et des fonctions linéaires par morceaux
  - des produits de variables

Variables avec domaines discontinues

- Que faire avec le cas où soit $x = 0$ OU $l <= x <= u$



- On peut considérer ceci comme deux contraintes, mais elles ne peuvent être vraies toutes les deux à la fois…

- Pouvez-vous trouver des exemples d'applications ?

- Comment modéliser ceci avec un PLNE ?

Variables avec domaines discontinues

- On utilisera une variable indicatrice:

$$y = \begin{cases} 0 & \text{for } x = 0 \\ 1 & \text{for } l \leq x \leq u \end{cases}$$

- Qu'on liera avec la variable originale par les contraintes suivantes:

$$x \leq uy$$
$$x \geq ly$$
$$y \quad \text{binary}$$

- *Y = 0* implique donc *x = 0* et *y = 1* implique que *l <= x <= u*

Les coûts fixes

- Soit le problème suivant:

Minimize: $C(x)$

Subject to:

$$a_i x + \sum_{j \in J} a_{ij} w_j \gtreqless b_i \qquad \forall i \in I$$

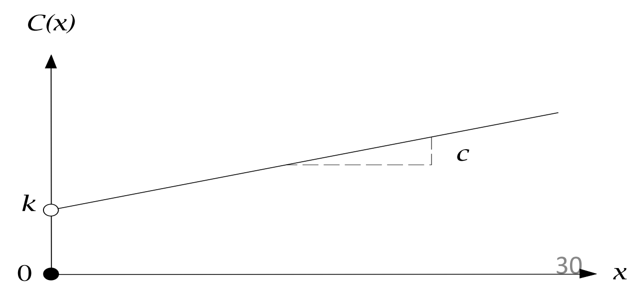
$$x \geq 0$$

$$w_j \geq 0 \qquad \forall j \in J$$

Where:

$$C(x) = \begin{cases} 0 & \text{for } x = 0 \\ k + cx & \text{for } x > 0 \end{cases}$$

- La fonction de coût n'est ni linéaire ni continue...
- À quelle application pensez-vous ?
- Comment résoudre ce problème ?



Les coûts fixes

- Si on connaît une borne *u* suffisamment grande pour *x* et qu'on introduit une variable indicatrice *y*

$$y = \begin{cases} 0 & \text{for } x = 0 \\ 1 & \text{for } x > 0 \end{cases}$$

- On relier *x* et *y* par *x <= yu*
- L'objectif devient donc:

$$C^*(x, y) = ky + cx$$

- Et le problème devient:

Minimize: $ky + cx$

Subject to:

$$a_i x + \sum_{j \in J} a_{ij} w_j \gtrless b_i \qquad \forall i \in I$$

$$x \leq uy$$

$$x \geq 0$$

$$w_j \geq 0 \qquad \forall j \in J$$

$$y \quad \text{binary}$$

Une disjonction de contrainte

- Soit le problème suivant:

Minimize:

$$\sum_{j \in J} c_j x_j$$

Subject to:

$$\sum_{j \in J} a_{1j} x_j \leq b_1 \qquad (1)$$

$$\sum_{j \in J} a_{2j} x_j \leq b_2 \qquad (2)$$

$$x_j \geq 0 \qquad \forall j \in J$$

- Où soit (1) ou (2) doit être respectée
- Des applications ?
- Comment faire ?

Une disjonction de contraintes

- Encore une fois on introduira une variable supplémentaire *y* ainsi que deux grands nombres (M_1 et M_2).
- En modifiant (1) et (2) de la manière suivante:

$$(1) \quad \sum_{j \in J} a_{1j} x_j \leq b_1 + M_1 y$$

$$(2) \quad \sum_{j \in J} a_{2j} x_j \leq b_2 + M_2 (1 - y)$$

- On s'assure qu'une des deux contraintes devra être satisfaite.

Minimize: $$\sum_{j \in J} c_j x_j$$

Subject to:

$$\sum_{j \in J} a_{1j} x_j \leq b_1 + M_1 y$$

$$\sum_{j \in J} a_{2j} x_j \leq b_2 + M_2 (1 - y)$$

$$x_j \geq 0 \qquad \forall j \in J$$

$$y \quad \text{binary}$$

Contraintes conditionnelles

- Une variante de ce problème survient lorsque certaines contraintes sont conditionnelles:

$$\textit{If} \quad (1) \quad \Big(\sum_{j\in J} a_{1j}x_j \le b_1\Big) \quad \textit{is satisfied,}$$
$$\textit{then} \quad (2) \quad \Big(\sum_{j\in J} a_{2j}x_j \le b_2\Big) \quad \textit{must also be satisfied.}$$

- Donnez des exemples d'application ?

- Comment traiter ce cas ?

Contraintes conditionnelles

- Pour adresser ce cas, nous devons nous tourner vers la logique

- L'équation logique qui nous intéresse est (A implique B)
- Cette équation est équivalente à (non-A OU B)
- On a donc une disjonction de contraintes, qu'on peut traiter comme précédemment…

$$\textit{If} \quad \Big(\sum_{j \in J} a_{1j} x_j \leq b_1\Big) \quad \textit{holds, then} \quad \Big(\sum_{j \in J} a_{2j} x_j \leq b_2\Big) \quad \textit{must hold,}$$

- Devient:

$$\Big(\sum_{j \in J} a_{1j} x_j > b_1\Big) \quad \textit{or} \quad \Big(\sum_{j \in J} a_{2j} x_j \leq b_2\Big) \quad \textit{must hold.}$$

- Sauf qu'ici on a un signe > qu'on ne peut traiter en PL…

Contraintes conditionnelles

- On introduira une petite valeur *epsilon* $\sum_{j \in J} a_{1j} x_j \geq b_1 + \epsilon$

- Pour obtenir: $\sum_{j \in J} a_{1j} x_j \geq b_1 + \epsilon, \quad or \quad \sum_{j \in J} a_{2j} x_j \leq b_2 \quad$ *must hold.*

- Qui peut être réécrit comme:

$$\sum_{j \in J} a_{1j} x_j \geq b_1 + \epsilon - Ly$$

$$\sum_{j \in J} a_{2j} x_j \leq b_2 + M(1 - y)$$

Les SOS (Special Ordered Sets)

- Considérer le cas particulier suivant:
  - Votre modèle comporte une série de décision oui/non ordonnée.
  - Seulement une décision « oui » est possible
  - Entre deux décisions « oui » on préfèrera toujours celle qui est la première dans la série.
- Pour ce cela, on dispose généralement d'une série de variables booléennes y_i telles que $\sum_i y_i \leq 1$
- On peut généraliser ce cas à :
  - Aux variables générales x_i tel que $0 <= x_i <= u$ et sujet à $\sum_i a_i x_i \leq b$
- On peut aussi considérer le cas où deux décisions « oui » sont permises, mais elles doivent être consécutives.
- Les solveurs ont des objets de modélisation SOS1 et SOS2 qui implémentent ces conditions de manières plus efficaces lors du branch and bound.

Fonctions linéaires par morceaux

- Soit le problème suivant:

Minimize:

$$\sum_{j \in J} f_j(x_j)$$

Subject to:

$$\sum_{j \in J} a_{ij} x_j \gtrless b_i \qquad \forall i \in I$$

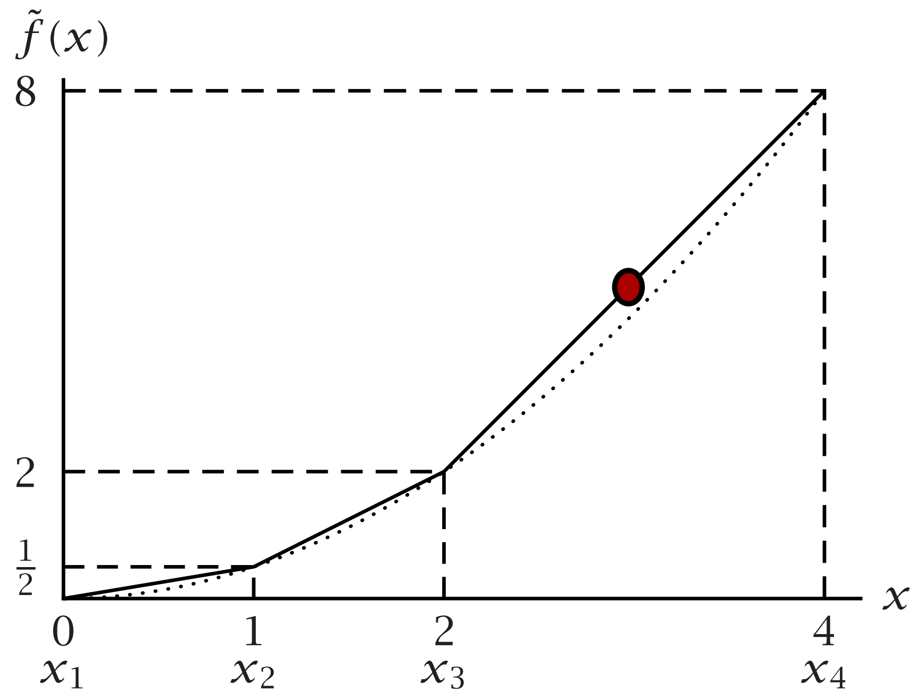
$$x_j \geq 0 \qquad \forall j \in J$$

- Ici on remarque que l'objective, bien que non linéaire, est séparable.
- C'est-à-dire que l'objectif est une somme de fonctions définies sur une variable à la fois.

Séparable	Non séparable
$x_1^2 + 1/x_2 - 2x_3 = f_1(x_1) + f_2(x_2) + f_3(x_3)$	$x_1 x_2 + 3x_2 + x_2^2 = f_1(x_1, x_2) + f_2(x_2)$
$x_1^2 + 5x_1 - x_2 = g_1(x_1) + g_2(x_2)$	$1/(x_1 + x_2) + x_3 = g_1(x_1, x_2) + g_2(x_3)$

Fonctions linéaires par morceaux

- Soit la fonction $f(x) = \frac{1}{2}x^2$



- Soit x_1, x_2, x_3, x_4 des points de « cassure » (breakpoints) auquel on évalue la fonction f(x) (soit 0,1,2,4)
- On approxime donc linéairement tout point situé entre deux cassures, par exemple f(3) = ½*f(2) + ½*f(4) = ½*1 + ½*8 = 5.

Fonctions linéaires par morceaux

- Une des manières de traiter ces fonctions est d'utiliser la λ-formulation.
- Soit λ_1, λ_2, λ_3, λ_4, 4 poids non négatifs dont la somme = 1, alors la fonction linéaire par morceaux précédente peut-être exprimée par:

$$\begin{aligned}
\lambda_1 f(x_1) + \lambda_2 f(x_2) + \lambda_3 f(x_3) + \lambda_4 f(x_4) &= \tilde{f}(x) \\
\lambda_1 x_1 + \lambda_2 x_2 + \lambda_3 x_3 + \lambda_4 x_4 &= x \\
\lambda_1 + \lambda_2 + \lambda_3 + \lambda_4 &= 1
\end{aligned}$$

- Comme au plus deux λ peuvent être non négatifs, et que ceux-ci doivent en plus être consécutifs, on peut ajouter une contrainte SOS2(λ).
- La majorité des solveurs ont un objet « Piecewise Linear » que vous pouvez utiliser directement.

Éliminer les produits de variables

- Que faire des problèmes où des termes contiennent le produit de deux variables booléennes x_1x_2

- On peut faire disparaître ce produit en introduisant une nouvelle variable booléenne y qui doit être égale au produit x_1x_2.

- Pour ce faire il faut ajouter les contraintes suivantes:

$$
\begin{aligned}
y &\leq x_1 \\
y &\leq x_2 \\
y &\geq x_1 + x_2 - 1 \\
y &\quad \text{binary}
\end{aligned}
$$

Éliminer les produits de variables

- Que faire maintenant si doit traiter un produit $x_1 x_2$ où x_1 est une variable binaire et x_2 est une variable continue tel que $0 <= x_2 <= u$?

- On introduit une variable continue y définie comme $y = x_1 x_2$ en ajoutant les contraintes ci-dessous imposant le comportement:

$$y \leq u x_1$$

$$y \leq x_2$$

$$y \geq x_2 - u(1 - x_1)$$

$$y \geq 0$$

x_1	x_2	$x_1 x_2$	**constraints**	**imply**
0	$w : 0 \leq w \leq u$	0	$y \leq 0$ $y \leq w$ $y \geq w - u$ $y \geq 0$	$y = 0$
1	$w : 0 \leq w \leq u$	w	$y \leq u$ $y \leq w$ $y \geq w$ $y \geq 0$	$y = w$

Excercise de modélisation

- On veut assembler l'horaire hebdomadaire de travail d'une unité d'infirmières sachant que:
  - On a trois quarts de 8h de travail à couvrir (J,S,N)
  - Le nombre d'infirmières requises de d_q où q est un quart de travail (J,S,N)
  - Une infirmière doit avoir 16h de repos entre deux quarts.

- Les contraintes suivantes doivent être respectées
  - Si on emploie une infirmière pendant la semaine, elle doit travailler au moins 3 quarts.
  - Une infirmière doit avoir congé soit
    - a) toutes les nuits de la semaine
    - b) toute la fin de semaine.
  - Si une infirmière travaille 4 quarts de travaille, elle doit avoir congé la fin de semaine.
  - Chaque infirmière donne une liste de jours de congé souhaitée (et ordonnées), on doit lui donner au moins un de ces choix.

- Il faut minimiser les coûts sachant que:
  1. Si on emploie une infirmière il faut payer l'agence un montant fixe de F$ + G$ par quart travaillé.
  2. En plus, il est permis de ne pas avoir le bon nombre d'infirmières, mais il faut payer une pénalité de P(k) $ ou k est la différence entre le nombre souhaité le nombre d'infirmières assignées