Programmation linéaire
Heure 2
Programmation linéaire en nombres entiers

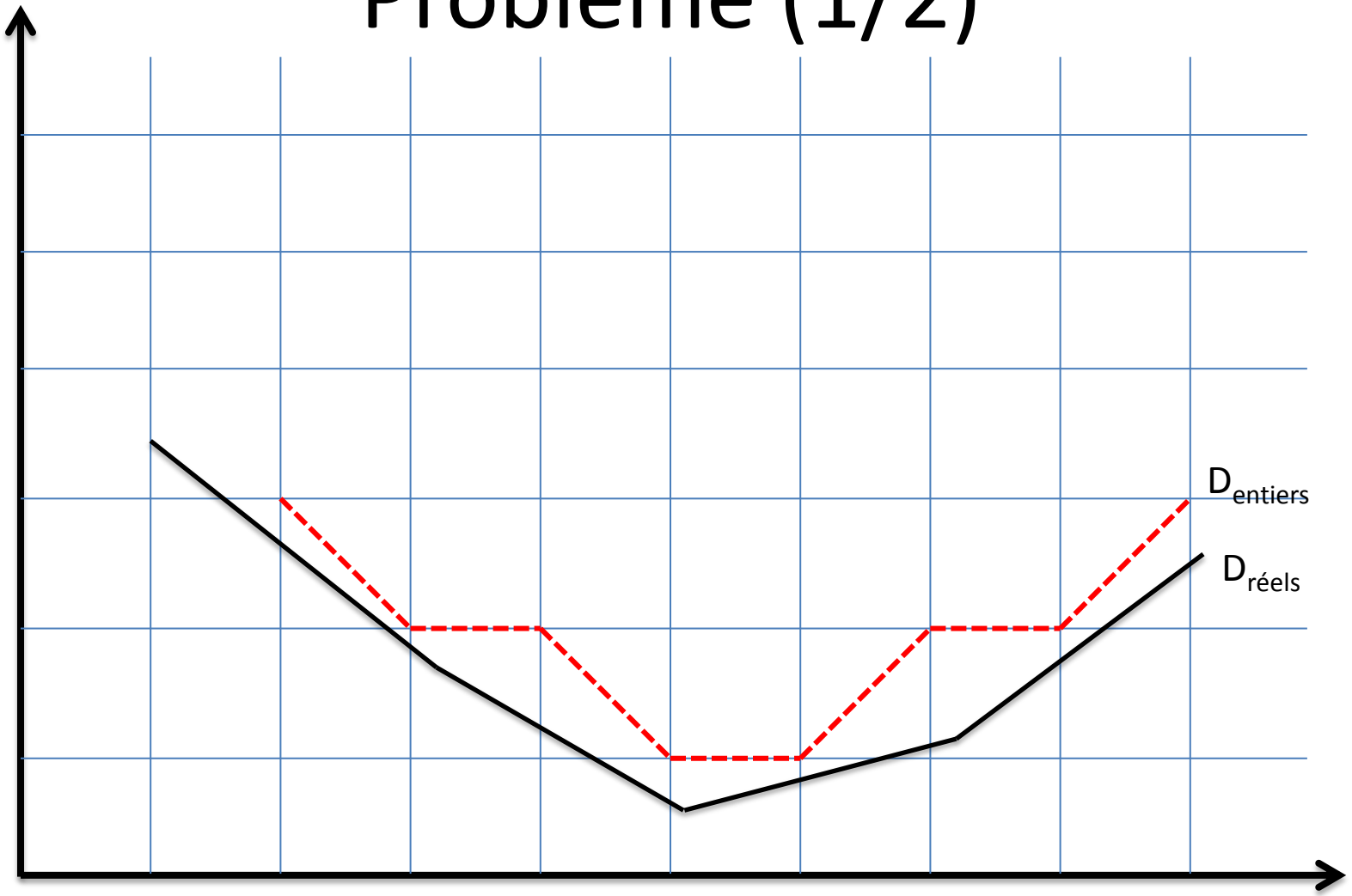
Philippe Morignot, Adrien Pain
pmorignot@yahoo.fr

Formulation

- L'algorithme du simplexe donne une solution sur l'ensemble des réels R.
- Comment faire si l'on se pose un problème en variables **entières** et non réelles ?

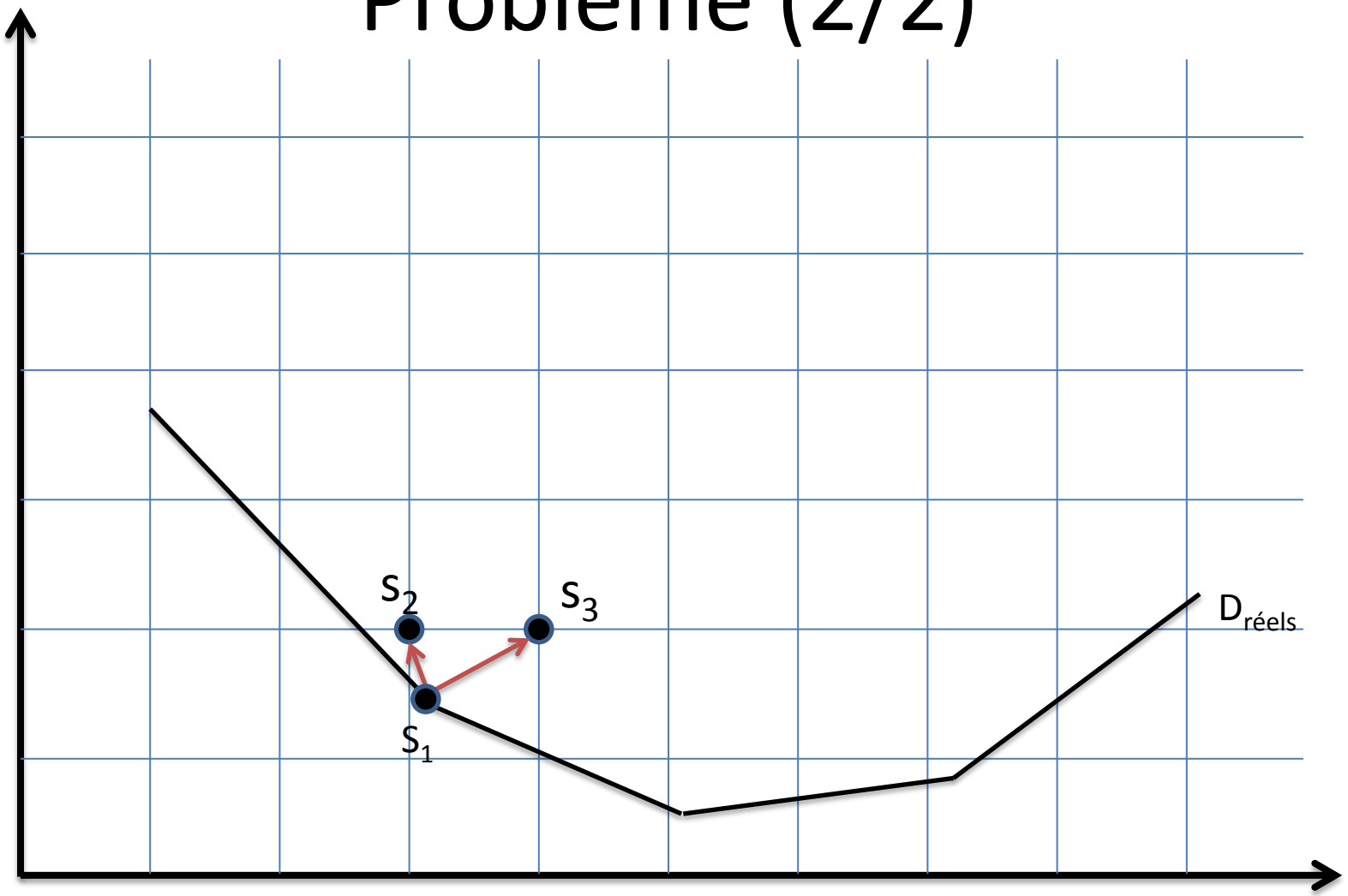
$$(x_1, x_2, \ldots, x_n) \in N \text{ au lieu de } R \text{ ?}$$

Problème (1/2)



Le domaine n'est plus convexe

Problème (2/2)



A partir de S_1, faut il aller en S_2 ou S_3 ?

Passage de réels aux entiers

- **Avec des variables entières :**
  - Le domaine n'est pas forcément convexe.
  - La solution sur R peut ne pas appartenir au domaine entier.
- **Avec des variables entières, les problèmes linéaires possibles sont :**
  - *Programmation Linéaire en Nombres Entiers* (PLNE) :
    $(x_1, x_2, ..., x_n) \in N$
  - *Programmation Linéaire en Variables Booléennes* (PLVB) :
    $(x_1, x_2, ..., x_n) \in \{0, 1\}$
  - *Programmation Linéaire en Variables Mixtes* (PLVM) :
    $(x_1, x_2, ..., x_p) \in R$
    $(x'_1, x'_2, ..., x'_q) \in N$
    $(x''_1, x''_2, ..., x''_r) \in \{0, 1\}$

Algorithme B&B

- **Algorithme de séparation / évaluation (en anglais : *branch-and-bound*) :**
  - Développer un arbre.
  - Chaque nœud de l'arbre est le problème de la racine + des contraintes de borne sur les variables.
  - Sur chaque nœud, on évalue la relaxation continue du problème, pour avoir une borne supérieure de z (si z doit être maximisé).
  - Parcours de l'arbre pour trouver l'optimum.

Exemple

- **Soit le problème P :**

$$\max (4x_1 - x_2) \text{ tel que :}$$

$$\begin{cases} 7x_1 - 2x_2 \leq 14 \\ x_2 \leq 3 \\ 2x_1 - 2x_2 \leq 3 \\ x_1, x_2 \geq 0 \\ x_1, x_2 \in N \end{cases}$$

- Soit P' le problème P, mais sur R et non sur N. P' est la **relaxation continue** de P.

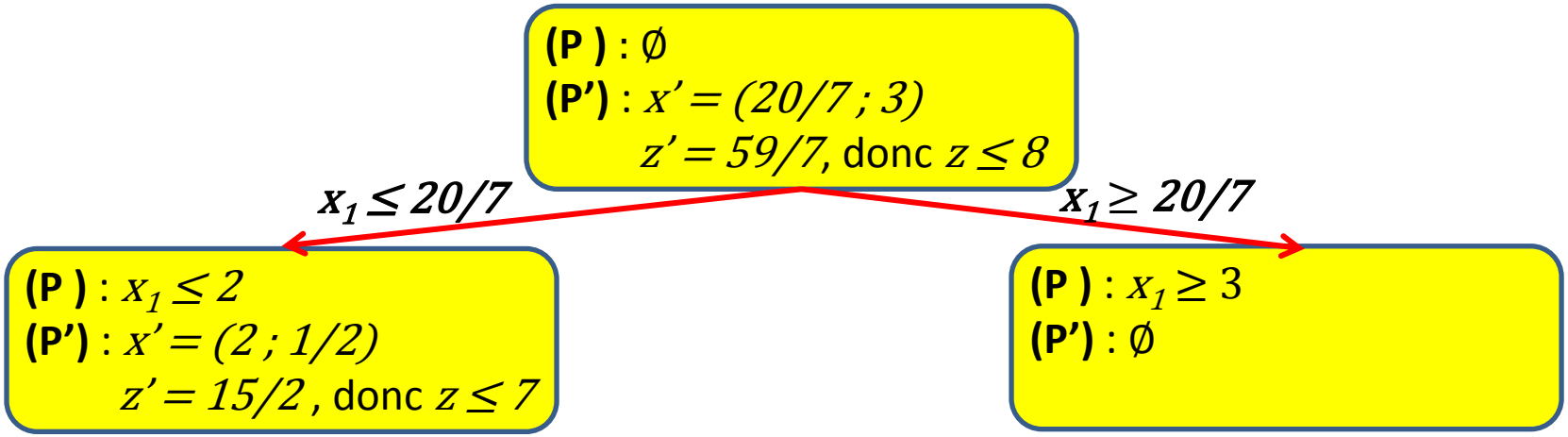
Algorithme de branch-and-bound (1/6)

- **Evaluer** la borne supérieure de la solution optimale de (P) par résolution de (P') :
  - Résoudre le problème en variables réelles (P') donne une borne supérieure de la solution de (P).
  - Solution réelle : $x' = (20/7, 3)$ et $z' = 59/7$
  - Borne supérieure de (P) = 8 $(56/7 < 59/7)$
- **Séparer** :
  - Considérer les deux possibilités : une variable entière est **supérieure** à la valeur réelle, ou **inférieure** à la valeur réelle.
- Développer récursivement un arbre.

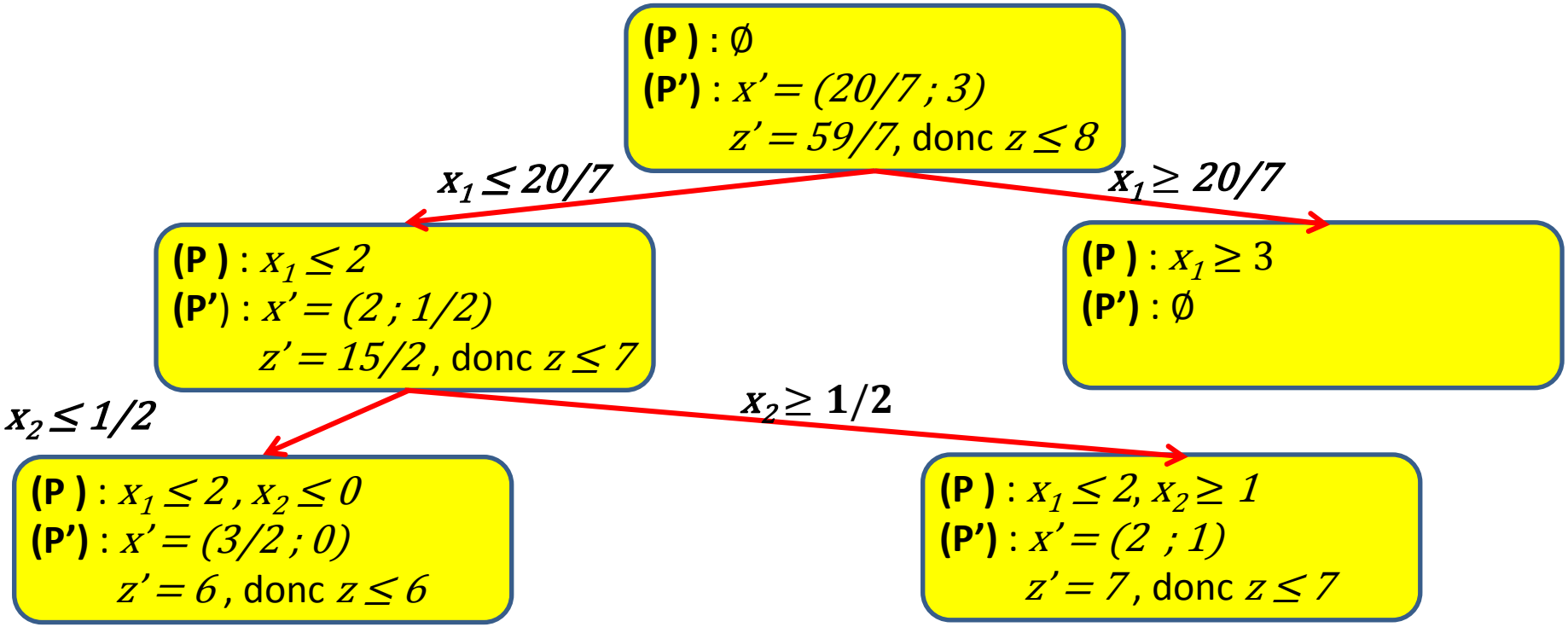
Algorithme de branch-and-bound (2/6)

(P) : $\emptyset$
(P') : $x' = (20/7 ; 3)$
$z' = 59/7$, donc $z \leq 8$

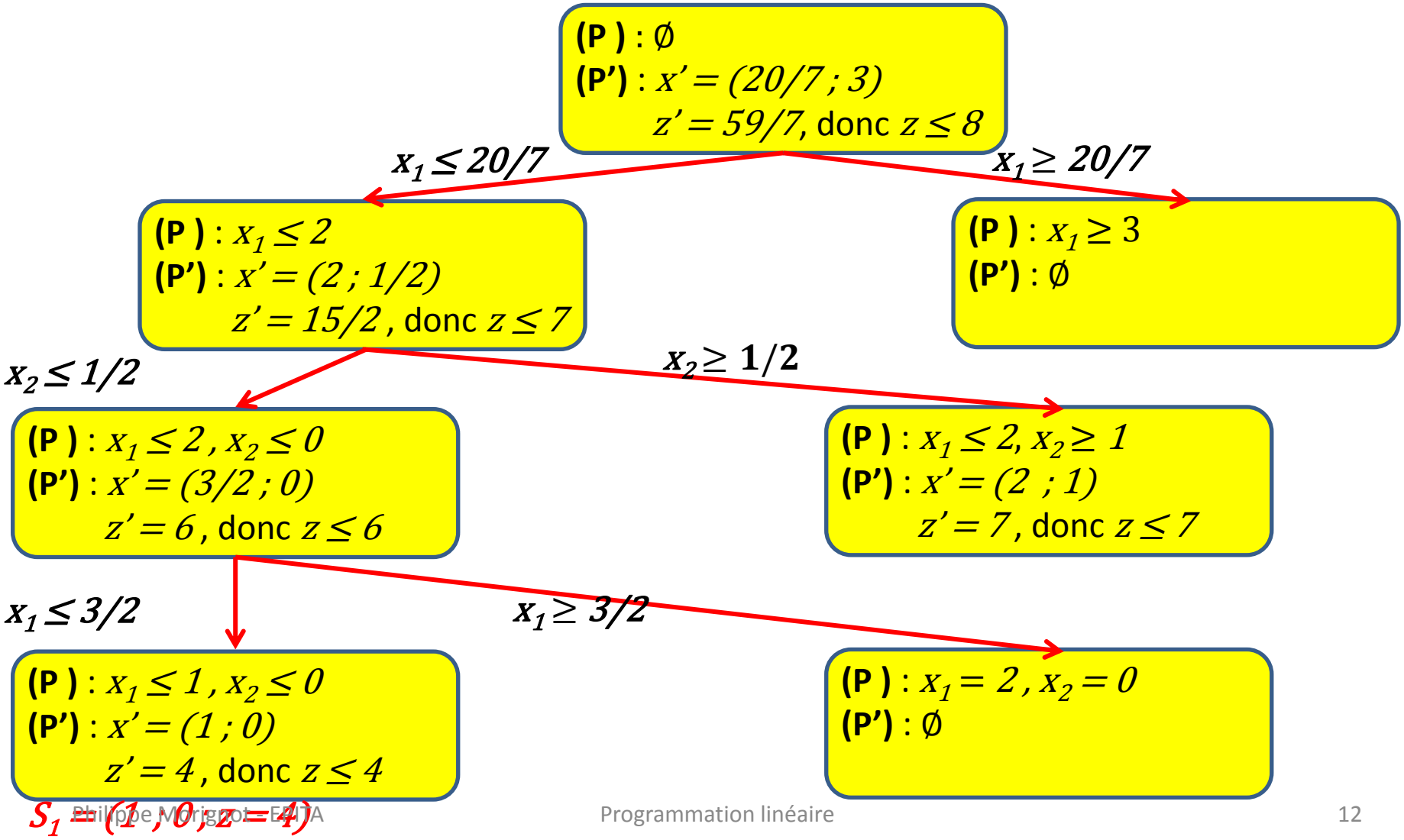
Algorithme de branch-and-bound (3/6)



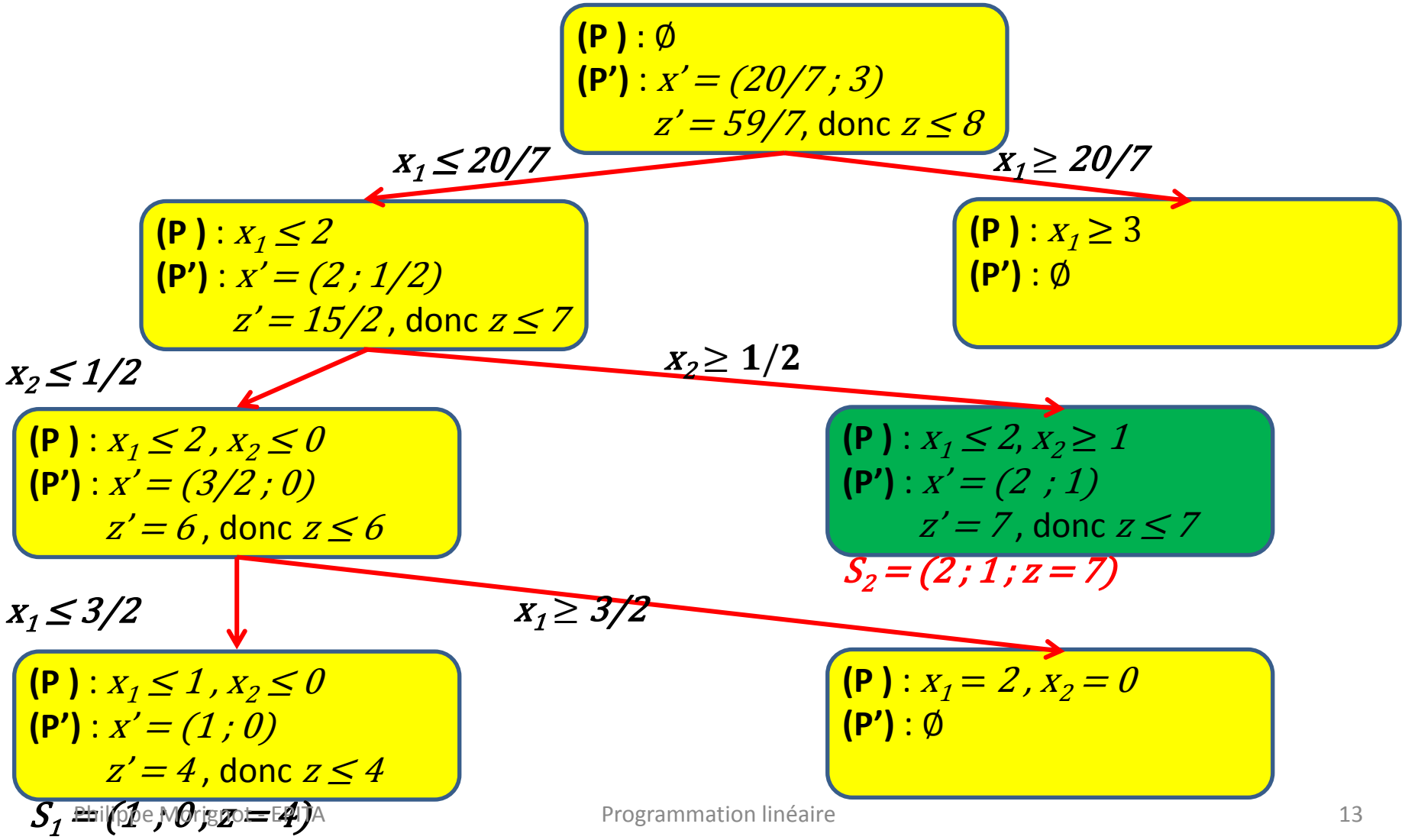
Algorithme de branch-and-bound (4/6)



Algorithme de branch-and-bound (5/6)



Algorithme de branch-and-bound (6/6)



Algorithme de branch-and-bound Pseudo-code

Procédure BRANCH-AND-BOUND

ENTREE : (P) : maximisation d'un programme linéaire en nombres entiers

SORTIE : (x', z') *:* solution optimale de (P)

1 $x' = \emptyset$

2 $z' = -\infty$

3 $problemes = \{P\}$

4 TANT QUE $problemes \neq \emptyset$ FAIRE

5 Choisir p dans $problemes$

6 Calculer $x'(p)$ et $z'(p)$ *avec* SIMPLEXE() // Solution relâchée

7 SI $x'(p) = \emptyset$ ou $z'(p) \leq z'$ ALORS continue // Non réalisable ou sous-optimale

8 SI $x'(p) \in N^*$ ALORS $x' \leftarrow x'(p)$ et $z' \leftarrow z(p)$ et continue // Solution entière

9 POUR TOUT $x'_i(p)$ non entière dans $x'(p)$ FAIRE // Séparation

10 $problemes \leftarrow problemes \cup \{ p \text{ et } x_i \leq x'_i(p) \} \cup \{ p \text{ et } x_i \geq x'_i(p) \} \setminus p$

11 return (x', z')

Heuristiques pour la fonction CHOISIR ligne 5

- Choisir le nœud avec le maximum en z.
- Choix de la variable suivante :
  - **Statique :** l'ordre de parcours des variables est fixé.
  - **Dynamique :** l'ordre de parcours des variables est déterminé en cours de parcours.
- Arrêter à un certain pourcentage de z_{init} : « assez bon ».
- Profondeur d'abord, largeur d'abord, etc.

Méthode de modélisation

1. Que veulent dire les indices ?
2. Quelles sont les variables de décision ?
3. Quelles sont les données du problème ?
4. Quelle est la fonction objectif/coût ?
5. Quelles sont les contraintes ?

Problème de transport (1/2)

- Plusieurs clients, plusieurs dépôts.
- Un client peut être livré par plusieurs dépôts.
- Chaque dépôt possède une capacité de stockage.
- Chaque client demande une certaine quantité.
- Transporter une unité de produit d'un dépôt à un client possède un certain coût.

Problème de transport (2/2)

1. **Indices :** i = i^e client ; j = j^e dépôt.
2. **Variables :** $x_{i,j}$ = quantité livrée au client i via le dépôt j.
3. **Données :**

   S_j = quantité disponible dans le dépôt j (stock)

   D_i = demande du client i

   $c_{i,j}$ = coût de transport d'une unité de marchandise entre le dépôt j et le client i
4. **Fonction de coût :** $z = min(\sum_{i,j} c_{i,j} x_{i,j})$
5. **Contraintes :**

   $\forall i, \ \sum_j x_{i,j} \geq D_i$ // Satisfaction de la demande

   $\forall j, \ \sum_i x_{i,j} \leq S_j$ // Capacité maximum des dépôts

   $x_{i,j} \in N$

Problème de transport augmenté

- **Nouvelle contrainte :** on ne veut livrer que des multiples de 100 unités.
- **Modifications du modèle de transport :**
  - Variables : $x_{i,j}$ = nombre de lots livrés au client i via le dépôt j
  - Données : Q = taille des lots à livrer (par ex., 100).
  - Remplacer $x_{i,j}$ par $Q\,x_{i,j}$ dans la fonction de coût et les contraintes précédentes.

Problème de chalandise

- Comme pour un problème de transport, mais un client n'est affecté qu'à un seul dépôt.
- **Variables :** $x_{i,j} \in \{0, 1\} = 1$ ssi le dépôt j livre le client i.
- **Données :** idem.
- **Fonction de coût :** $z = min\left(\sum_{i,j} c_{i,j} D_i x_{i,j}\right)$
- **Contraintes :**

$\forall i, \sum_j x_{i,j} = 1$ // Satisfaction de la demande.

$\forall j, \sum_i D_i x_{i,j} \leq S_j$ // Capacité maximum des dépôts.

$x_{i,j} \in \{0, 1\}$ // Si [0, 1], problème de transport.

Problème de transport avec choix des dépôts

- Comme pour un problème de transport, avec en plus :
  - un dépôt peut être ouvert ou fermé
  - l'ouverture d'un dépôt j possède un coût k_j (N_{max} dépôts ouverts).
- **Variables :**

  $x_{i,j}$: la quantité $x_{i,j}$ va du dépôt j au client i.

  $y_j \in \{0, 1\} = 1$ ssi le dépôt j est ouvert.
- **Fonction de coût :** $z = min\,(\sum_{i,j} c_{i,j}\, x_{i,j} + \sum_j k_j\, y_j)$
- **Contraintes :**

$\forall\, i,\ \sum_j x_{i,j} \geq D_i$ // Satisfaction de la demande

$\forall\, j,\ \sum_i x_{i,j} \leq S_j y_j$ // Capacité maximum des dépôts ouverts

$\sum_j y_j \leq N_{max}$ // Nombre de dépôts ouverts

$x_{i,j} \in N \ \ et \ \ y_j \in \{0, 1\}$

Conclusion

- En programmation linéaire en nombres entiers, les variables prennent leur valeur dans N et non dans R.
- Algorithme de branch-and-bound : recherche arborescente avec relaxation continue à chaque nœud et séparation de chaque nœud en deux nœuds-fils. Considérer les deux cas à chaque nœud.
- Plusieurs heuristiques pour parcourir l'arbre du B&B.
- Exemples de problèmes en PLNE : transport, transport augmenté, chalandise et transport avec choix des dépôts.