

# Recherche Opérationnelle

Cédric BENTZ  
Maître de conférences UPS

M1 MIAGE  
2011-2012

<http://www.lri.fr/~bentz/supports/RO.html>

# La RO, c'est quoi ?

- Recherche Opérationnelle (RO) :
  - Discipline récente (environ 60-70 ans) et comportant de nombreux aspects différents
  - A la frontière entre informatique, économie et mathématiques appliquées
  - Une tentative de définition (Source : ROADEF)
    - « La Recherche Opérationnelle (RO) est la discipline des méthodes scientifiques utilisables pour élaborer de meilleures **décisions**. Elle permet de **rationaliser**, de simuler et d'**optimiser** l'architecture et le fonctionnement des systèmes de production ou d'organisation. »

# Les origines de la RO

- De nombreux problèmes historiques, parfois très anciens, peuvent être modélisés, puis résolus, comme des problèmes de RO :
  - Carrés magiques (Chine antique)
  - Les 7 ponts de Königsberg (L. Euler, vers 1750)
  - Problème des 8 reines (F. Nauck, vers 1850)
  - Problème du voyageur de commerce (vers 1850)
  - Problème des 4 couleurs (19e)

# Histoire récente de la RO (1/2)

- Origines militaires de la RO
  - Objectif : optimisation de la logistique militaire et du réapprovisionnement des troupes
    - Création par le gouvernement britannique du « *Committee for Operations Research* » au début de la seconde guerre mondiale
    - G. Dantzig, « conseiller scientifique » à la *U.S. Air Force*, met au point en 1947 l'algo. du simplexe
  - Nécessite de puissants centres de calcul
  - A l'origine, le mot *opérationnelle* signifie donc *en vue de programmer des opérations (militaires)*
- *R* de *RO* = *recherche* de solutions concrètes

# Histoire récente de la RO (2/2)

- Depuis, le domaine de la RO s'est étendu à des applications « civiles », en particulier grâce à :
  - L'essor des connaissances dans le domaine
  - L'amélioration de la puissance de calcul des CPU
- Dans ce cadre, d'autres aspects possibles de la prise de décision ont alors été introduits :
  - Le décideur doit tenir compte de plusieurs critères
  - Toutes les données ne sont pas connues de façon certaine, et comportent donc une part d'incertitude

# RO, modèles et décision

- La RO permet de « meilleures » décisions
  - Outils : modèles formels/mathématiques, permettant de représenter le cadre général de la décision à prendre
  - Points essentiels :
    - Le modèle doit être établi en liaison avec le décideur final, et **tous les aspects nécessaires** à la prise de décision doivent être présents dans ce modèle (comme pour un modèle UML)
    - Les variables du modèle doivent donc refléter la décision à prendre : résoudre le problème ainsi modélisé (c'est-à-dire affecter les bonnes valeurs aux variables du modèle) permet alors immédiatement d'en déduire la décision associée
      - Variables = quelle décision doit-on prendre (solution à trouver) ?
      - Critère(s)/préférence(s) = dans quel(s) but(s)/avec quel(s) objectif(s) ?
      - Contraintes = les conditions impératives que doit vérifier la décision

# La RO en pratique (1/3)

- Résolution de problèmes de décision complexes
  - Problèmes de grandes tailles
  - Aspects combinatoire, multicritère, stochastique...
- Plusieurs sous-domaines identifiables :
  - Programmation mathématique
  - Optimisation combinatoire (aspect combinatoire)
  - Aide à la décision (dont aspect multicritère)
  - Optimisation stochastique (aspect aléatoire)
  - Et beaucoup d'autres...

# La RO en pratique (2/3)

- Sert à résoudre tous types de problèmes difficiles (issus de l'industrie ou de la finance) :
  - Problèmes de télécoms : conception, exploitation, fiabilité, routage (Orange Labs – ex. FT R&D)
  - Problèmes de transport (DRT SNCF, Air France)
  - Production et transport d'énergie (EDF, GDF)
  - Planification de projets/plannings (EURODECISION)
  - Ingénierie financière, composition optimale de portefeuilles d'actions (banques...)
  - Et beaucoup d'autres impliquant une utilisation optimale de ressources limitées (modèles PL/PLNE)

# La RO en pratique (3/3)

- Démarche « classique » de résolution de tels problèmes par un utilisateur de la RO
  - Modéliser le problème sous une forme **exploitable**
    - Cette phase de modélisation préalable est absolument **indispensable**, et sa bonne réalisation est donc critique
  - Appliquer la ou les méthodes de résolution idoines disponibles en catalogue, selon la forme du modèle
    - Beaucoup sont implémentées dans des « solveurs » (logiciels de résolution), commerciaux ou non, et peuvent donc être aisément intégrées à une autre application
    - La recherche en RO, elle, s'attache précisément à améliorer les méthodes de résolution du catalogue

# Objectifs du cours (1/2)

- Exposer les résultats majeurs et principales techniques associés à deux domaines :
  - Programmation linéaire (= *Optimisation* linéaire)
  - Optimisation combinatoire (OC)  
*(Pas d'aspect multicritère, ni stochastique...)*
- Deux aspects en particulier seront abordés
  - Apprendre à modéliser un problème sous une forme « pertinente », facilitant sa résolution
  - Connaître les techniques adaptées à la résolution efficace de ces problèmes

# Objectifs du cours (2/2)

- La phase de résolution pouvant être confiée à des *solveurs*, pourquoi étudier les méthodes de résolution dans le cadre de ce cours ?

# Objectifs du cours (2/2)

- La phase de résolution pouvant être confiée à des *solveurs*, pourquoi étudier les méthodes de résolution dans le cadre de ce cours ?
  - ✗ Parce que c'est dans le programme du cours, et qu'il faut donc bien le faire ?

# Objectifs du cours (2/2)

- La phase de résolution pouvant être confiée à des *solveurs*, pourquoi étudier les méthodes de résolution dans le cadre de ce cours ?
  - ✗ Parce que c'est dans le programme du cours, et qu'il faut donc bien le faire ?
  - ✗ Parce que c'est inutile en pratique, et qu'il est donc absolument indispensable de le voir en cours ?

# Objectifs du cours (2/2)

- La phase de résolution pouvant être confiée à des *solveurs*, pourquoi étudier les méthodes de résolution dans le cadre de ce cours ?
  - ✗ Parce que c'est dans le programme du cours, et qu'il faut donc bien le faire ?
  - ✗ Parce que c'est inutile en pratique, et qu'il est donc absolument indispensable de le voir en cours ?
  - ✓ Car les idées sur lesquelles sont basées ces méthodes permettent de mieux comprendre certaines particularités de ces problèmes, et d'en fournir une autre interprétation, souvent éclairante...

# Organisation du cours

- Format de l'UE :
  - 24h de cours : C. B.
  - 26h de TD : C. B., L. Montero
- Evaluation :
  - CC (partiel) = 33% de la note
  - Examen final sur table = 67% de la note
- Pré-requis :
  - Maths de base (pré-bac), notions d'algorithmique

# Plan du cours (1/3)

- Programmation linéaire (PL)
  - Définitions
  - Modélisation
    - Mise sous forme standard, linéarisation
  - Résolution graphique
  - Algorithme du simplexe :
    - Polyèdres, sommets et enveloppes convexes
    - Description à l'aide de dictionnaires
    - Dégénérescences et cyclage
    - Initialisation (méthode des 2 phases)
  - Dualité

# Plan du cours (2/3)

- Programmation linéaire entière (PLNE)
  - Définitions
  - Modélisation
    - Utilisation de variables entières, linéarisation(s)
  - Résolution (exacte) d'un PLNE
    - Relaxations, bornes, liens avec la PL
    - Méthodes par Séparation & Evaluation, Coupes
  - Résolution (approchée) d'un PLNE
    - Heuristiques, méta-heuristiques
  - Cas « faciles » et totale unimodularité

# Plan du cours (3/3)

- Problèmes de chemins et de flots
  - Chemins optimaux
    - Algorithmes : plus courts chemins, plus longs chemins
    - Application des chemins optimaux : modèles d'ordonnancement de projets
  - Flots maximums
    - Modèle PL(NE)
    - Algorithme de Ford & Fulkerson (variante avec graphes d'écart)
    - Dualité Flot / Coupe
    - Flots maximums à coût minimal
    - Application : le problème d'affectation linéaire

# Bibliographie (en français)

- **Ouvrages en français (pour débiter) :**
  - ***Méthodes d'optimisation combinatoire.***  
*I. Charon, Anne Germa, O. Hudry. (MASSON).*
  - ***Précis de recherche opérationnelle.*** R. Faure, B. Lemaire, C. Picouleau. (DUNOD).
- **Ouvrages en français (plus complets) :**
  - ***Programmation mathématique : théorie et algorithmes.*** M. Minoux. (DUNOD).
  - ***Optimisation combinatoire - programmation discrète.*** M. Sakarovitch. (HERMANN)

# Bibliographie (en anglais)

- **Ouvrage en anglais (pour débiter) :**
  - *Linear Programming*. V. Chvatal. (FREEMAN).
- **Ouvrages en anglais (plus complets) :**
  - *Integer Programming*. L. Wolsey. (WILEY).
  - *Theory of linear and integer programming*. A. Schrijver. (WILEY).

# Programmation Linéaire

Première partie :

Le modèle de la  
Programmation Linéaire  
et sa résolution graphique

# La programmation linéaire (PL) en 2 mots et 2 slides (1/2)

- La PL est un des domaines de la RO :
  - Origines militaires de la PL : G. Dantzig, conseiller scientifique à la *U.S. Air Force*, met au point la méthode du simplexe pour résoudre des PL en 1947
  - Le terme « programmation » fait ici référence à l'« organisation » d'opérations militaires et à leur « planification » (et non au sens informatique usuel)
  - Aussi appelé **Optimisation linéaire**, car on optimise une fonction linéaire sous des contraintes linéaires
  - Cas particulier de la programmation mathématique (fonction à optimiser et contraintes quelconques)

# La programmation linéaire (PL) en 2 mots et 2 slides (2/2)

- PL = technique de résolution de problèmes :

Modélisation par un Programme Linéaire (PL) :

- Modèle assez générique,
- Permet de capturer de nombreux problèmes concrets.



Résolution par des algo. dédiés efficaces en théorie et/ou en pratique

- Plus efficaces que des algorithmes génériques de Prog. Math.,
- Par ex. : algorithme du simplexe, points intérieurs, ellipsoïdes, etc.

Solveurs rapides maintenant disponibles, basés sur ces algorithmes

- Par exemple : CPLEX, LPSolve, Xpress, Excell...
- Intégration aisée dans tout type d'applications (Java, C++, C, C#...)

# Un exemple introductif (1/5)

- Un brasseur doit décider de son plan de fabrication de bière. Il peut fabriquer :
  - Bière blonde (prix de vente : 15 euros par UV)
  - Bière brune (prix de vente : 25 euros par UV)
- 3 ingrédients sont à disposition, présents en quantités différentes dans les deux bières :
  - Maïs
  - Houblon
  - Malt

# Un exemple introductif (2/5)

- Quantités requises (par UV) :
  - Bière blonde : 2.5 kg de maïs, 125 g de houblon, 17.5 kg de malt
  - Bière brune : 7.5 kg de maïs, 125 g de houblon, 10 kg de malt
- Le brasseur dispose, après achat, de ces quantités de matières premières (MP) :
  - 240 kg de maïs
  - 5 kg de houblon
  - 595 kg de malt

# Un exemple introductif (3/5)

- Problématique du brasseur ?
  - Maximiser son bénéfice
  - Calcul du bénéfice ?
    - Revenus de la vente des 2 types de bières - Prix d'achat des matières premières
- Il « suffit » donc de maximiser son revenu
  - Comment ? Autrement dit, quelles quantités (en UV) de chaque bière fabriquer ?

# Un exemple introductif (4/5)

- Formulation du problème
  - **Variables**
    - $x_1 \geq 0$  est le nb d'UV de bière blonde fabriquées
    - $x_2 \geq 0$  est le nb d'UV de bière brune fabriquées
  - **Objectif** : maximiser le revenu
    - Revenu = revenu b. blonde + revenu b. brune
  - **Contraintes** de quantité sur les MP disponibles
    - Utiliser au plus 240 kg de maïs (en kg)
    - Utiliser au plus 5 kg de houblon (en g)
    - Utiliser au plus 595 kg de malt (en kg)

# Un exemple introductif (5/5)

- Modèle mathématique obtenu (PL)

$\max 15 x_1 + 25 x_2$  } **Maximiser** une fonction **linéaire** en  $x_1$  et  $x_2$

sous contraintes :

$$x_1 + 3 x_2 \leq 96 \text{ (maïs) //en divisant par 2.5}$$

$$x_1 + x_2 \leq 40 \text{ (houblon) //en divisant par 125}$$

$$7 x_1 + 4 x_2 \leq 238 \text{ (malt) //en divisant par 2.5}$$

} 3 **contraintes linéaires**  
en  $x_1$  et  $x_2$

$$\underbrace{x_1 \geq 0, x_2 \geq 0}$$

**Deux variables réelles**  $x_1$  et  $x_2$ , contraintes à être **positives**

# Description formelle d'un PL

- Fonction **économique** / fonction **objectif** :

$$\max/\min \underbrace{c_1x_1 + c_2x_2 + \dots + c_nx_n}$$

$c_i$  = ième coefficient (réel) de la fonction économique

- $m$  **contraintes linéaires** = de la forme :

$$\underbrace{m_1x_1 + m_2x_2 + \dots + m_nx_n} \left\{ \begin{array}{l} \geq b \\ \leq b \\ = b \end{array} \right\} \begin{array}{l} b = \text{second} \\ \text{membre de} \\ \text{la contrainte} \end{array}$$

$m_i$  = ième coefficient (réel)  
de la contrainte

**Contraintes  
de borne :**  
 $x_i \geq 0, x_i \leq 5, \dots$

- Les contraintes et la fonction économique sont des **combinaisons linéaires** des  $n$  variables réelles  $x_i$

# Forme canonique (FC) d'un PL

- Objectif : **max**  $c_1x_1 + c_2x_2 + \dots + c_nx_n$

- Sous les contraintes :

$$m_{11}x_1 + m_{12}x_2 + \dots + m_{1n}x_n \leq \underline{b}_1$$

...

$$m_{m1}x_1 + m_{m2}x_2 + \dots + m_{mn}x_n \leq \underline{b}_m$$

$x_i \geq \underline{0}$  pour tout  $i$  entre 1 et  $n$

# Notation compacte pour la FC d'un PL

- Objectif :  $\max c_1x_1 + c_2x_2 + \dots + c_nx_n \Rightarrow \max c(x)$

- Sous les contraintes :

$$m_{11}x_1 + m_{12}x_2 + \dots + m_{1n}x_n \leq b_1$$

...

$$m_{m1}x_1 + m_{m2}x_2 + \dots + m_{mn}x_n \leq b_m$$

}  $\Rightarrow x \in A$   
( $A =$ **polyèdre**  
admissible)

$$x_i \geq 0 \text{ pour tout } i \text{ entre } 1 \text{ et } n$$

}  $\Rightarrow x \geq 0$

# Notation compacte pour le problème du brasseur

- Problème du brasseur (rappel)

$$\left. \begin{array}{l} - \max 15 x_1 + 25 x_2 \end{array} \right\} \Rightarrow \max c(x) = 15 x_1 + 25 x_2$$

$$x_1 + 3 x_2 \leq 96 \text{ (maïs)}$$

$$x_1 + x_2 \leq 40 \text{ (houblon)}$$

$$7 x_1 + 4 x_2 \leq 238 \text{ (malt)}$$

$$x_1 \geq 0, x_2 \geq 0$$

$$\Rightarrow x \in A$$

# Propriété de la FC d'un PL

- **Tout PL peut être mis sous FC**
  - Contrainte  $m_1x_1 + m_2x_2 + \dots + m_nx_n \geq b \Leftrightarrow ?$
  - Contrainte  $m_1x_1 + m_2x_2 + \dots + m_nx_n = b \Leftrightarrow ?$
  - $x_i \leq 0 \Leftrightarrow ?$
  - $x_i$  réelle (non contrainte en signe)  $\Leftrightarrow ?$
  - Transformation de la fonction objectif
    - $\text{Min } c(x) \Leftrightarrow \text{Max } ?$  (réponse dans la suite...)
- Hyp. non restrictive pour la suite : PL sous FC

# Forme standard (FS) d'un PL

- Forme standard (FS) d'un PL ?
  - Idem forme canonique, **sauf que toutes les contraintes sont des égalités**
- Propriété : on peut passer de la FC d'un PL à sa FS en ajoutant à chaque contrainte une **variable d'écart**  $e_j \geq 0$  (car associée à un écart  $\geq 0$ )
  - On peut aussi retrancher une variable d'écart à toute contrainte de la forme  $\geq b$
- Conséquence : tout PL peut être mis sous FS !

# Petit lexique de la PL

- $x_1, x_2, \dots, x_n$  : variables de décision (par opposition aux **variables d'écart**)
- Solution **admissible** = affectation de valeurs aux  $x_i$  **vérifiant les contraintes**
- Région/ensemble/domaine/polyèdre **admissible** = ensemble des solutions admissibles
- Solution **optimale** = solution admissible qui maximise la fonction économique

# Optimalité d'une solution

- Soit  $A$  un polyèdre et  $c$  une fonction objectif
- Une solution  $x^*$  dans  $A$  est optimale pour le PL  $P = \{\max c(x), \text{ avec } x \in A\}$  si et seulement si **pour tout  $x$  dans  $A$ , on a  $c(x) \leq c(x^*)$** 
  - Transformation de  $\min c(x)$  ? De  $\max (-c(x))$  ?
- **Valeur optimale** de  $P = c(x^*)$ , avec  $x^*$  optimale
  - Droite d'équation  $c(x) = c(x^*)$  : **droite optimale**

# Linéarisation d'un problème

- Parfois, un programme mathématique PM peut être « **linéarisé** », c'est-à-dire mis sous la forme d'un PL **équivalent** au PM initial :
  - Intérêt : le résoudre à l'aide d'algorithmes de PL !
  - Ici, « équivalent » ne signifie rien d'autre que « qui a la même valeur optimale »
- Par exemple, comment linéariser :
  - Le PM  $\{\max \exp(c(x)), \text{ avec } x \in A\}$  ?
  - Le PM  $\{\max \min\{c(x), c'(x)\}, \text{ avec } x \in A\}$  ?

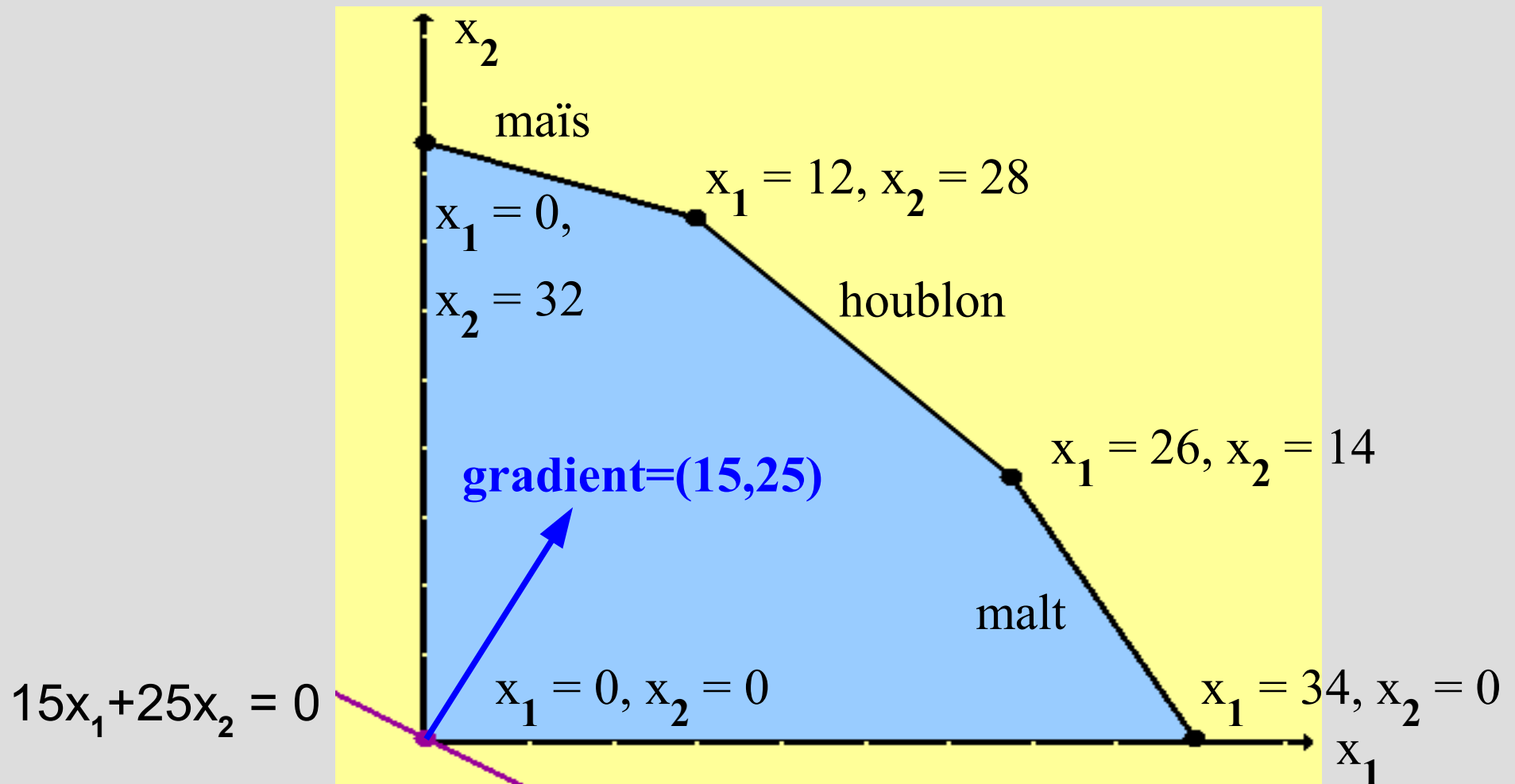
# Résolution graphique d'un PL

- Méthode dite de « **résolution graphique** »
  - Exploite le fait qu'on a 2 variables :  $x_1 \geq 0$  et  $x_2 \geq 0$
- Toute inégalité est vérifiée par une moitié du plan
  - Contrainte d'inégalité = demi-plan (dimension 2)
  - Frontière/bord d'un demi-plan = droite (dimension 2)
  - Tracé du domaine admissible dans le plan
    - Tracé du bord du demi-plan défini par chaque contrainte
    - Région admissible = intersection de tous ces demi-plans
  - Tracé d'une droite parallèle à la droite optimale
  - Recherche « visuelle » d'une solution optimale

# Résolution graphique du problème du brasseur (1/3)

- Dans le domaine admissible (en bleu), on cherche une solution qui maximise  $15x_1 + 25x_2$
- On trace la droite  $15x_1 + 25x_2 = 0$  (en violet)
  - La droite optimale est  $15x_1 + 25x_2 = ?$  (où  $? = val. opt.$ )
  - Cette droite est donc **parallèle** à  $15x_1 + 25x_2 = 0$
- Idée (visuelle) : « déplacer » la droite  $15x_1 + 25x_2 = 0$ 
  - La faire « glisser » le plus possible dans le bon sens
    - C'est-à-dire dans une direction qui augmente la valeur de la fonction objectif  $15x_1 + 25x_2$
    - Tout en restant dans le domaine admissible

# Résolution graphique du problème du brasseur (2/3)

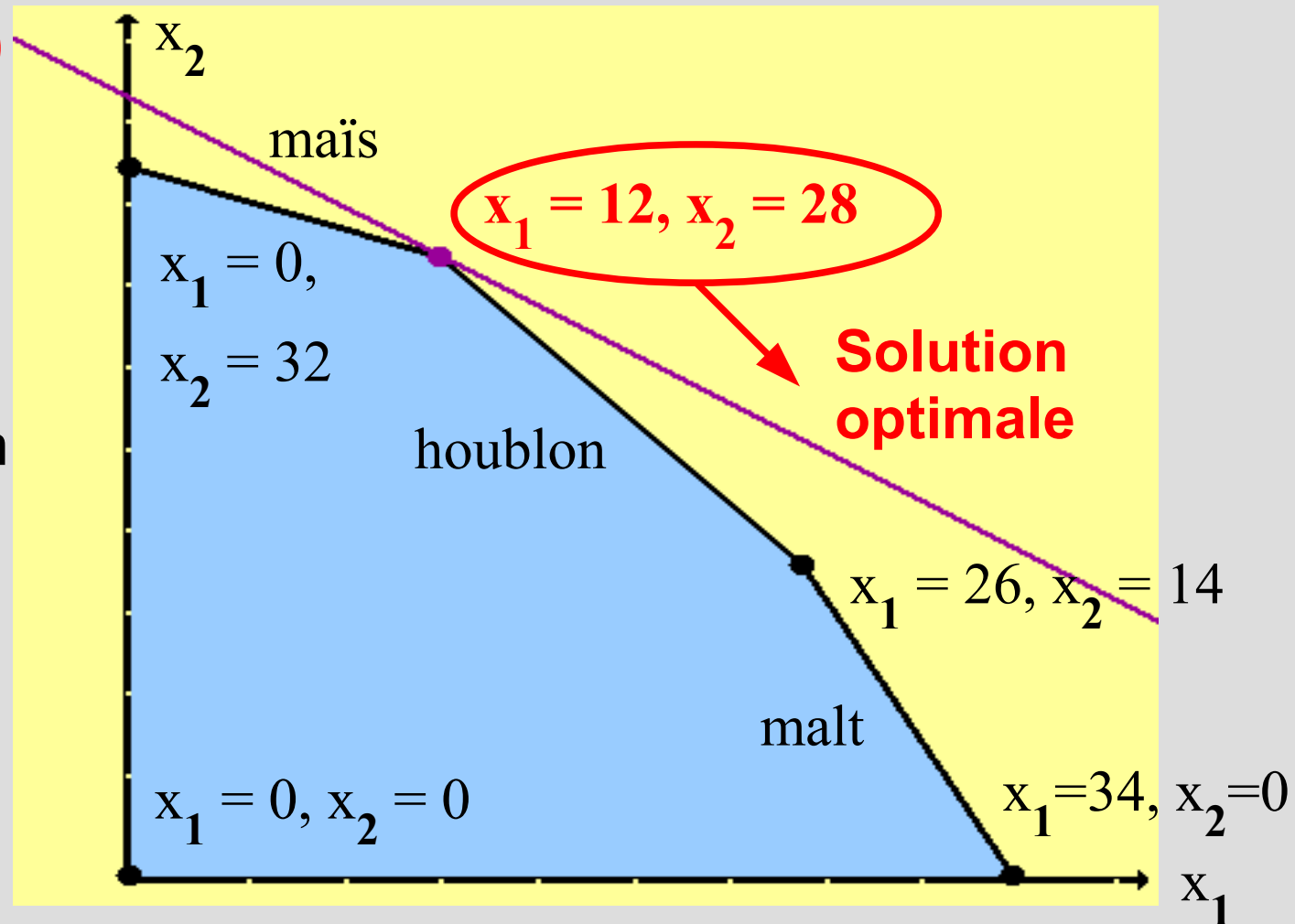


# Résolution graphique du problème du brasseur (3/3)

$$15x_1 + 25x_2 = 880$$

Droite optimale

**Plan de fabrication optimal (unique) :**  
12 UV de b. blonde,  
28 UV de b. brune,  
maïs épuisé,  
houblon épuisé,  
revenu = 880 euros

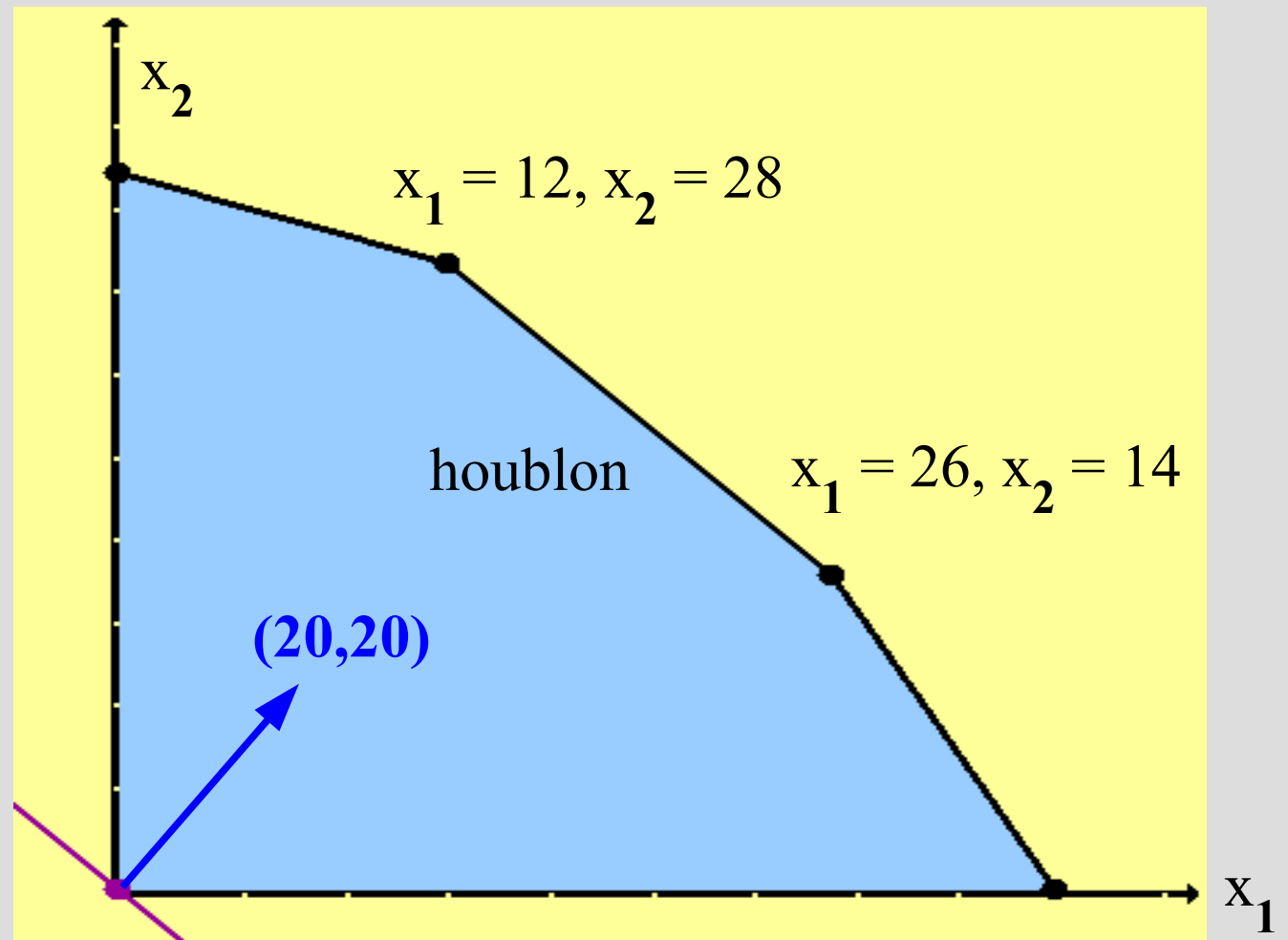


# Une deuxième variante (1/2)

- Si revenu b. blonde = revenu b. brune = 20 ?

Nouvelle fonction économique :  
 $\max 20 x_1 + 20 x_2$

$$20 x_1 + 20 x_2 = 0$$



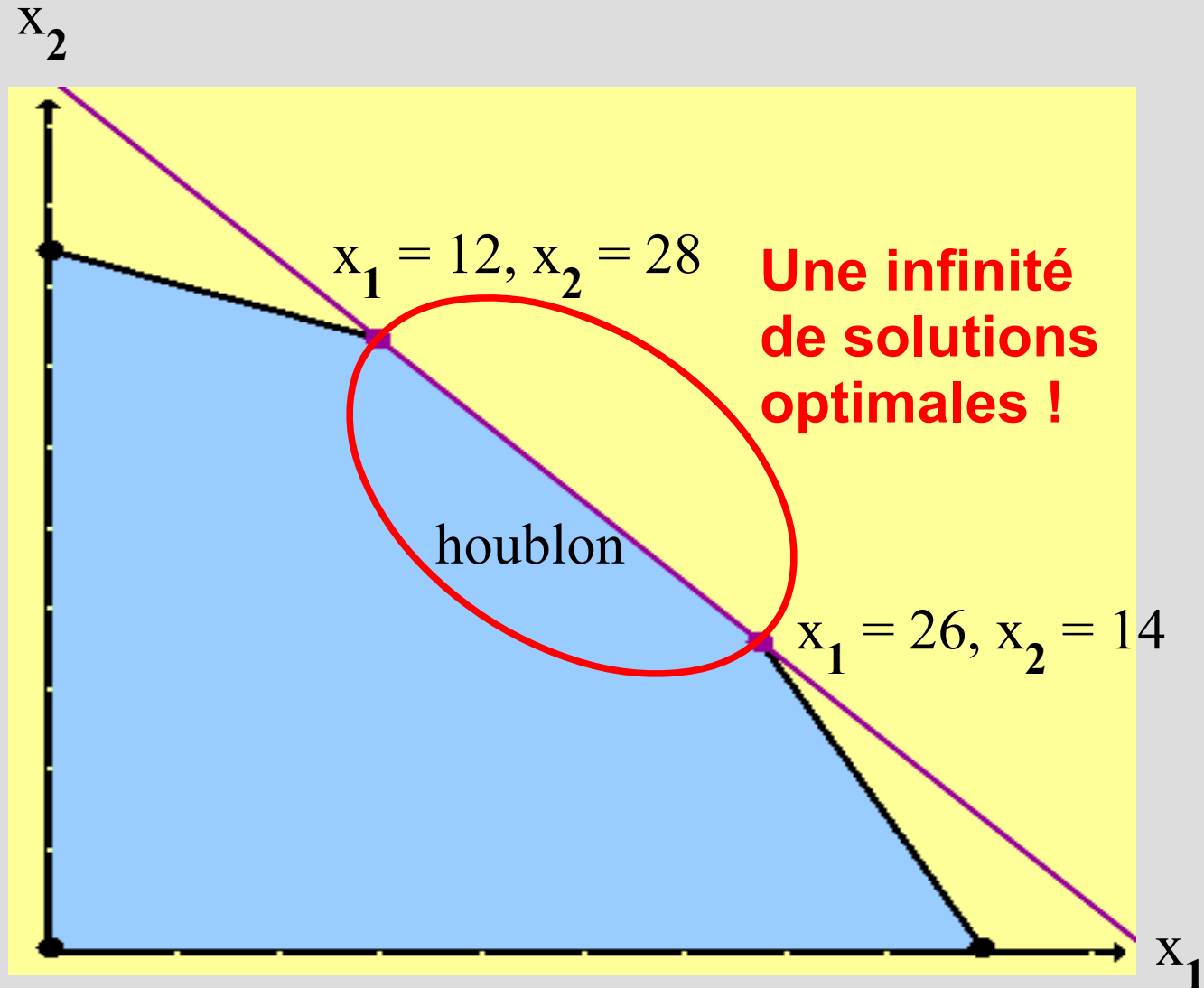
# Une deuxième variante (2/2)

$$20x_1 + 20x_2 = 800$$

Droite optimale

**Plans de fabrication optimaux :**

houblon épuisé,  
revenu = 800 euros  
(Par exemple :  
12 UV de b. blonde,  
28 UV de b. brune)



# Un premier bilan sur la PL (1/2)

- Ensemble admissible d'un PL ?
  - En dimension 2 ( $n=2$ ), région du plan bornée par des droites, c'est-à-dire polygone
  - En dimension quelconque, région de  $\mathbb{R}^n$  bornée par  $p$  **hyperplans** (**polyèdre**)
- Solutions optimales d'un PL ?

**Pas de solution admissible**



# Un premier bilan sur la PL (2/2)

- Si la valeur optimale est bornée, alors, que le domaine admissible soit borné ou non, on a :
  - Soit **une seule solution optimale**
    - En un sommet du polyèdre (polygone si  $n=2$ ) constituant la région admissible
  - Soit **une infinité de solutions optimales**
    - Si  $n=2$ , fonction objectif parallèle à l'une des contraintes
- Dernier cas : l'inverse est-il vrai ?
  - Non, il peut n'y avoir qu'une solution optimale, même si la fonction objectif est parallèle à une contrainte !

# Aller plus loin ?

- On sait maintenant résoudre des PL si  $n = 2$ 
  - On pourrait imaginer le faire avec  $n=3$  variables (résolution graphique en dimension 3)
  - Et avec  $n \geq 4$  variables ?
    - Résolution graphique **inapplicable**
    - Nécessité d'une **méthode algébrique** (et automatisée)
- A retenir...
  - PL = optimisation linéaire
  - Forme canonique, forme standard
  - Résolution graphique de PL à 2 variables (3 cas)

# Programmation Linéaire

Deuxième partie :

L'algorithme du simplexe

# Intuition « géométrique » derrière l'algorithme du simplexe

- Rappel : on veut résoudre des PL à  $n \geq 4$  variables
  - Méthode graphique inexploitable !
  - Difficulté initiale en PL : trouver une meilleure solution parmi un ensemble **infini** de solutions potentielles
  - Se ramener à un nombre **fini** de solutions ?
  - Idée simple conduisant à deux algorithmes :
    - Approche naïve (peu efficace)
    - Algorithme du simplexe

# Solution de base (concept non géométrique)

- Base d'un PL en FS à  $n$  variables et  $m$  contraintes
  - On suppose les contraintes linéairement indépendantes
  - **Base = choix de  $m$  variables**
    - Les  $m$  colonnes associées sont linéairement indépendantes (sous-matrice inversible)
  - **Les  $n-m$  autres variables sont nulles** ( $\Rightarrow \leq m \text{ var. } > 0$ )
- Solution de base d'un PL en FS (définition « light »)
  - Solution associée à une base de ce PL  $\Rightarrow$  exemple ?
  - Une telle solution  $x$  est **admissible** ssi  $x \geq 0$
  - Un PL en FC peut en avoir jusqu'à  $C_{n+m}^m = (n+m)! / (n!m!)$

# Convexité

- Un ensemble  $S \subset \mathbb{R}^n$  est **convexe** si pour tout  $x, y$  dans  $S$ , tout point du segment  $[x, y]$  est dans  $S$ 
  - Exemple ( $n=2$ ) : segment  $[x, y]$ , un carré, un disque...
- **Enveloppe convexe** d'un ensemble de points  $x_i$ 
  - C'est le plus petit ensemble convexe contenant les  $x_i$
  - Par exemple : triangle pour 3 points non alignés...

# Sommet d'un polyèdre (concept géométrique)

- Un point  $x$  d'un polyèdre  $P$  est un **sommet** (ou **point extrême**) ssi il n'est pas sur un segment reliant deux autres points de  $P$  entre eux
  - Ex. : sommets d'un segment  $[x,y]=\{x,y\}$
- **Propriété : un polyèdre borné est exactement l'enveloppe convexe de ses sommets**
  - Corollaire 1 : tout polyèdre borné est convexe
  - Corollaire 2 : tout PL qui a une (ou des) solution(s) optimale(s) a (au moins) un de ses sommets optimal
    - Vrai même si le domaine admissible est non borné...

# Approche « naïve » pour la PL

- Idée initiale simple :
  - Tout polyèdre de  $\mathbb{R}^n$  a un nombre **fini** de sommets
  - Calculer la valeur de la fonction objectif en chaque sommet et garder le meilleur d'entre eux ?
  - NON : méthode peu efficace, car on peut montrer que
    - **Pour un polyèdre  $P$  et pour  $x \in P$ ,  $x$  est un sommet de  $P$  ssi  $x$  est une solution de base admissible de  $P$**
    - Le PL du brasseur a donc 5 solutions de base admissibles
    - Si  $m=n=50$ , on peut avoir  $C_{n+m}^m = C_{100}^{50} \approx 10^{29} = 10^{12}(10^9 \cdot 10^8)$  solutions de base admissibles (impraticable) !

# Vers une approche moins naïve : idée de l'algorithme du simplexe

- Principe de l'**algorithme du simplexe** :
  - Plutôt que d'énumérer toutes les solutions de base admissibles, on va en parcourir certaines
  - Concrètement, on passe d'un sommet (solution de base admissible) à un autre, dont la valeur est meilleure (vis-à-vis de la fonction objectif)
  - Finalement, quand ce n'est plus possible, on peut montrer qu'on obtient bien une solution optimale
  - Il reste à trouver comment faire : la présentation qui suit ne nécessite même pas d'avoir compris les liens évoqués avec les solutions de base admissibles...

# Une autre vision de la FS : les dictionnaires

- Soit  $P = \{\max c(x), x \in A, x \geq 0\}$  un PL sous FS
  - On définit le **dictionnaire initial** de  $P$  en faisant passer toutes les variables à droite, sauf les variables d'écart
    - Donc, équivalence entre la FS et ce dictionnaire initial
  - Plus généralement, un **dictionnaire** est l'expression de  $m$  variables en fonction des autres ( $m = \text{nb contraintes}$ )
    - Les variables « à gauche » sont supposées non nulles, les variables « à droite » sont supposées nulles
    - Dictionnaire  $\leftrightarrow$  solution de base !
    - Le dictionnaire « initial » est donc associé à l'origine (variables de décision nulles, variables d'écart non nulles)

# Exemple de dictionnaire : le PL du brasseur

**Dictionnaire initial** de valeur 0 (associé aux variables d'écart  $x_3, x_4, x_5$ ) :  $x_3=96, x_4=40, x_5=238$

1

$$x_3 = 96 - x_1 - 3x_2$$

$$x_4 = 40 - x_1 - x_2$$

$$x_5 = 238 - 7x_1 - 4x_2$$

$$z = 3x_1 + 5x_2$$

2

$$x_2 = 32 - 1/3 x_1 - 1/3 x_3$$

$$x_4 = 8 - 2/3 x_1 + 1/3 x_3$$

$$x_5 = 110 - 17/3 x_1 + 4/3 x_3$$

$$z = 160 + 4/3 x_1 - 5/3 x_3$$

3

$$x_1 = 12 + 1/2 x_3 - 3/2 x_4$$

$$x_2 = 28 - 1/2 x_3 + 1/2 x_4$$

$$x_5 = 42 - 3/2 x_3 + 17/2 x_4$$

$$z = 176 - x_3 - 2x_4$$

Améliorer cette solution (admissible) ?

Idée : augmenter une seule des variables nulles (à droite), en améliorant  $z$

1. Choix de variable (dictionnaire initial) :  $x_2$  (choix arbitraire),  $x_1$  reste nul

2. Problème :  $x_1=0$ , mais de combien peut-on augmenter  $x_2$  pour que :

$$\left. \begin{array}{l} x_3 \geq 0 \Leftrightarrow 96 - 3x_2 \geq 0 \Leftrightarrow x_2 \leq 32 \\ x_4 \geq 0 \Leftrightarrow 40 - x_2 \geq 0 \Leftrightarrow x_2 \leq 40 \\ x_5 \geq 0 \Leftrightarrow 238 - 4x_2 \geq 0 \Leftrightarrow x_2 \leq 119/2 \end{array} \right\} \begin{array}{l} x_2 = 32 \\ x_3 = 0 \end{array}$$

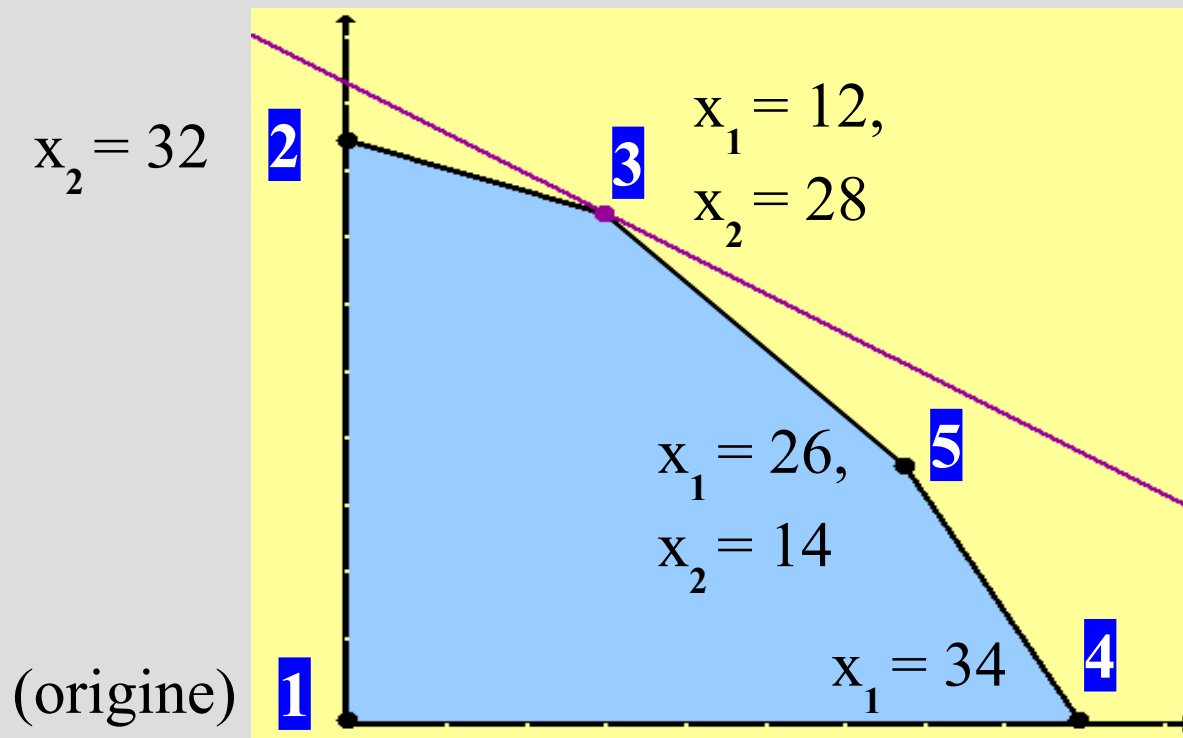
3. Recommencer (nouveau dictionnaire) ?

Exprimer  $z, x_2, x_4, x_5$  en fonction de  $x_1, x_3$

STOP, car  $z \leq 176$  et donc solution optimale !

# Bilan de la résolution du PL du brasseur par les dictionnaires

Durant le processus, 3 solutions admissibles (sommets) visitées



- 1 Montre que  $z \geq 0$
- 2 Montre que  $z \geq 160$
- 3 Montre que  $z \geq 176$   
(cette solution, de valeur  $5 \cdot 176 = 880$ , est optimale car  $176 - x_3 - 2x_4 \leq 176$ )

Si, dans le 1er dictionnaire, on augmente  $x_1$  et non  $x_2$ , on visite  $4 > 3$  solutions admissibles **1 4 5 3**  $\Rightarrow$  en général, quel est le meilleur chemin ? Sa taille ?

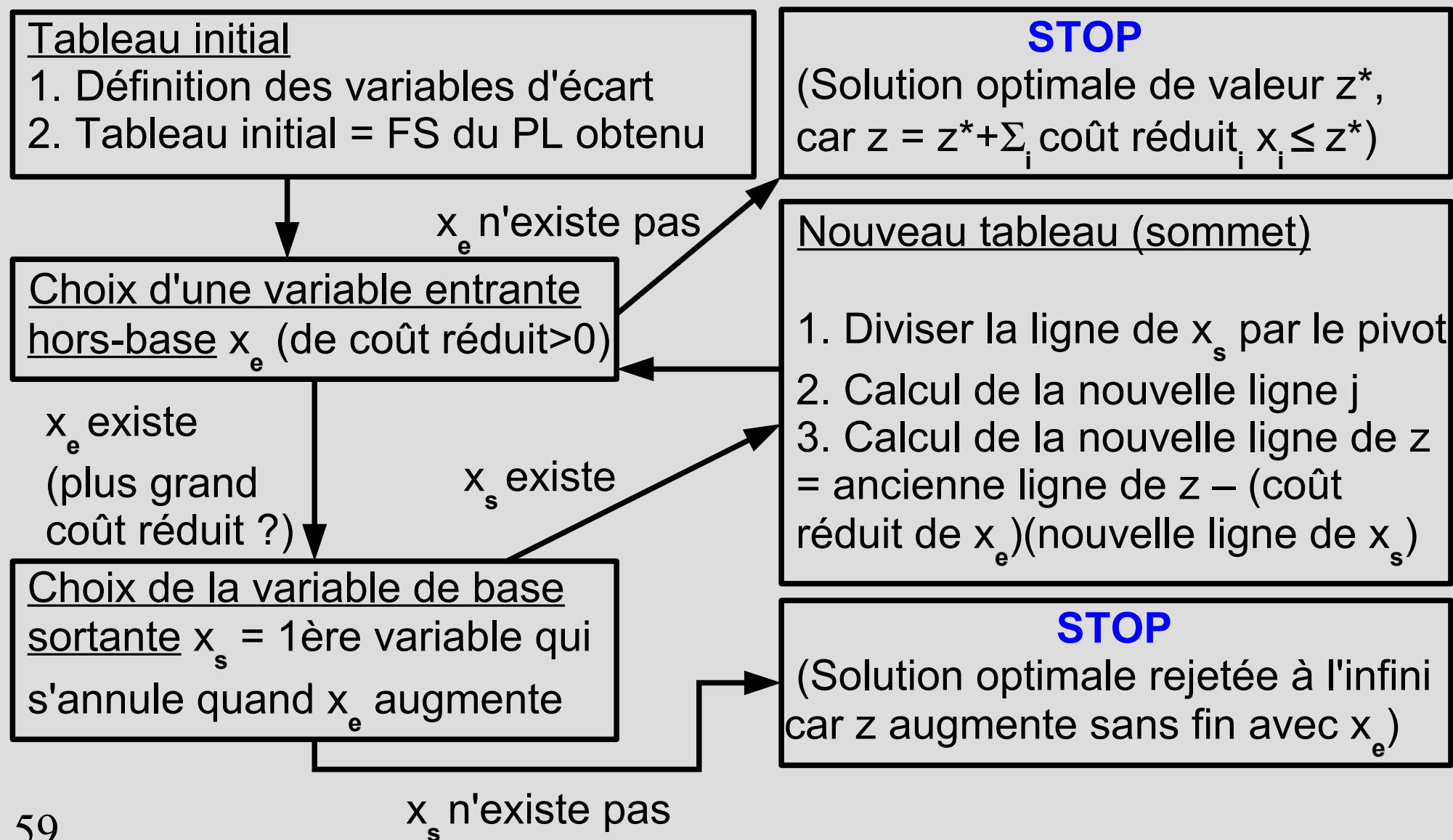
# Tableaux du simplexe

- **Tableau** = dictionnaire avec toutes les variables (nulles et non nulles) à gauche de l'égalité
  - Forme « traditionnelle » de l'algorithme du simplexe
  - Principe identique aux dictionnaires
    - Sur chaque ligne : une variable non nulle exprimée (donc de coefficient 1) en fonction des variables nulles
    - Condition d'optimalité identique : ligne de z quasi-inchangée (gauche  $\leftrightarrow$  droite), seul le terme constant change de côté
    - Comme pour les dictionnaires, 1 tableau=1 solution de base
    - Formules de mises à jour (= passage de la solution courante à la solution suivante) ?

# Vocabulaire des tableaux

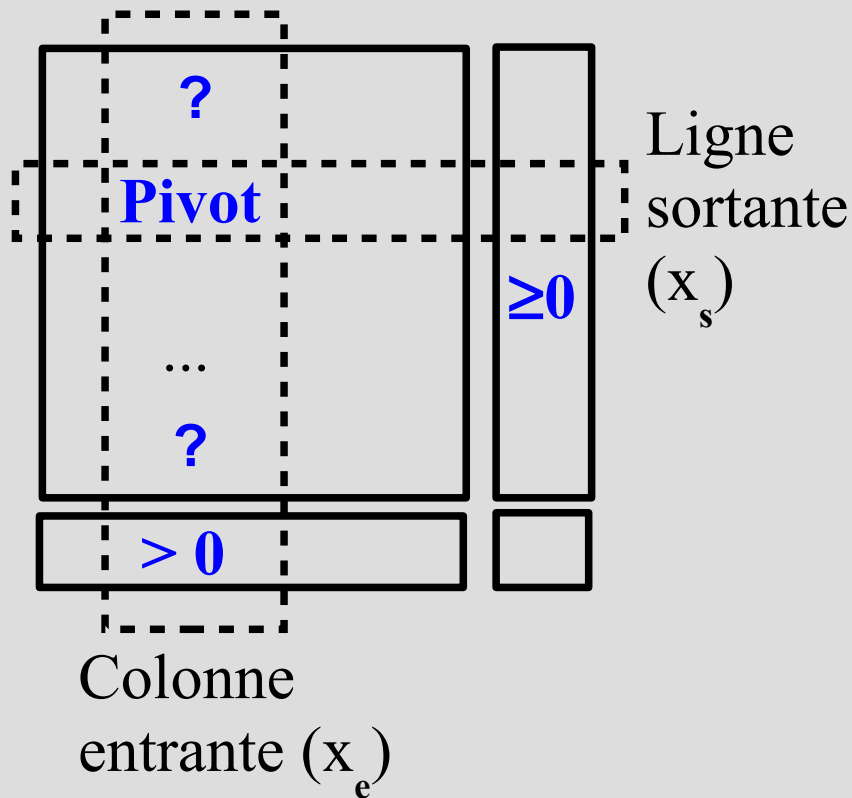
- Variable :
  - **hors-base** = variable nulle
  - **en base** = variable non nulle
  - **entrante** = variable nulle qui devient non nulle
  - **sortante** = variable non nulle qui devient nulle
- **Coûts réduits** = coefficients dans l'expression de la fonction objectif  $z$
- **Pivot** = coefficient de la variable entrante dans l'expression de la variable sortante

# Résolution de PL sous FC à l'aide de tableaux

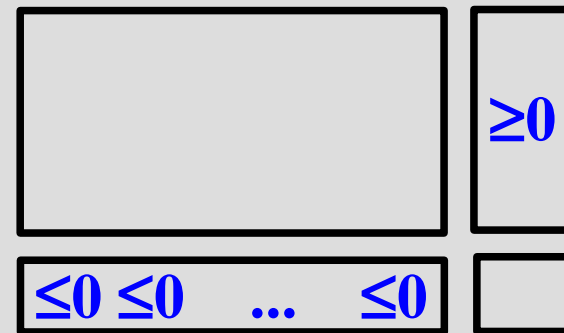


# Illustration de la méthode des tableaux du simplexe

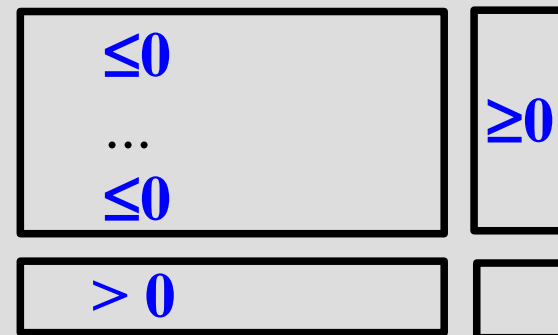
Cas général



Cas particulier 1 : la solution est optimale



Cas particulier 2 : solution optimale rejetée à l'infini



# Une infinité de solutions optimales

- Rappel : un PL ayant une valeur optimale finie peut admettre une infinité de solutions optimales
  - Condition nécessaire et suffisante (cf PL brasseur) ?
    - Cas  $n=2$  : une contrainte parallèle à la fonction objectif, qui définit un des côtés « bien situés » de la région admissible
  - Et en terme de tableaux du simplexe ?
    - A une itération associée à un des sommets situés sur cette contrainte, une des variables hors-base a un coût réduit nul
      - Augmenter cette variable ne modifie donc pas la valeur de l'objectif
    - D'où CNS : infinité de solutions optimales ssi une solution optimale avec (au moins) un coût réduit hors-base nul

# Effacité du simplexe

- Rappel : plusieurs « cheminements » possibles pour aboutir à la solution optimale par simplexe
  - Comment trouver le meilleur ? Et sa longueur ?
    - Cf PL du brasseur : deux, de longueurs respectives 3 et 4
  - Problème : en théorie, l'algorithme peut être obligé de visiter un nb de sommets non polynomial en  $m$  et  $n$ 
    - Cf Klee & Minty
    - Mais il finit : à chaque itération, on explore un sommet meilleur, donc au pire on explore tous les sommets (nb fini)
      - En fait, l'algorithme peut parfois ne pas finir : cf dégénérescences...
    - En pratique, il est très utilisé car très efficace : **entre  $3m/2$  et  $3m$  itérations en moyenne** ( $m$  = nombre de contraintes)

# Instances de Klee & Minty

- Instances où le simplexe fonctionne très mal
  - Les bonnes performances pratiques de l'algorithme tendent à indiquer que ce genre d'instances est rare
  - $2^n$  itérations nécessaires =  $2^n$  sommets visités
    - Choix variable entrante ?

Forme générale

$$\max \sum_{j=1}^n 10^{n-j} x_j$$

sous contraintes :

$$(2 \sum_{j=1}^{i-1} 10^{i-j} x_j) + x_i \leq 100^{i-1},$$

pour  $1 \leq i \leq n$

$$x_i \geq 0, \text{ pour } 1 \leq i \leq n$$

Pour le cas  $n=2$

$$\max \{10 x_1 + x_2, \text{ sous contraintes : } x_1 \leq 1, 20 x_1 + x_2 \leq 100, x_1 \geq 0, x_2 \geq 0\}$$

Quatre sommets visités :

1.  $(x_1=0, x_2=0)$ , de valeur 0
2.  $(x_1=1, x_2=0)$ , de valeur 10
3.  $(x_1=1, x_2=80)$ , de valeur 90
4.  $(x_1=0, x_2=100)$ , solution optimale de valeur 100

# Une première complication : les dégénérescences

- En général, l'algorithme du simplexe termine car
  - A chaque itération, la valeur de la fonction objectif est strictement augmentée (solution meilleure)
  - Ainsi, on n'explore pas deux fois chaque sommet
- Cette stricte augmentation provient du fait qu'une variable nulle (hors-base) devient  $> 0$  (en base)
  - En cas de **dégénérescence**, toutes les variables d'une même ligne peuvent être nulles
    - De telles solutions ont donc plus de  $n-m$  variables nulles
    - La terminaison de l'algo. du simplexe est alors compromise

# Solutions dégénérées et cyclage

- Un tableau correspond à une **solution dégénérée**
  - S'il y a au moins un 0 dans le 2<sup>nd</sup> membre du tableau
  - De façon équivalente, si au moins une variable de base est nulle dans la solution associée au tableau
- Non seulement la preuve de terminaison de l'algorithme du simplexe est remise en cause par le phénomène de dégénérescence, mais en fait il peut **effectivement** empêcher sa terminaison
  - Phénomène de **cyclage** : l'algorithme peut tourner indéfiniment sans jamais trouver une solution optimale

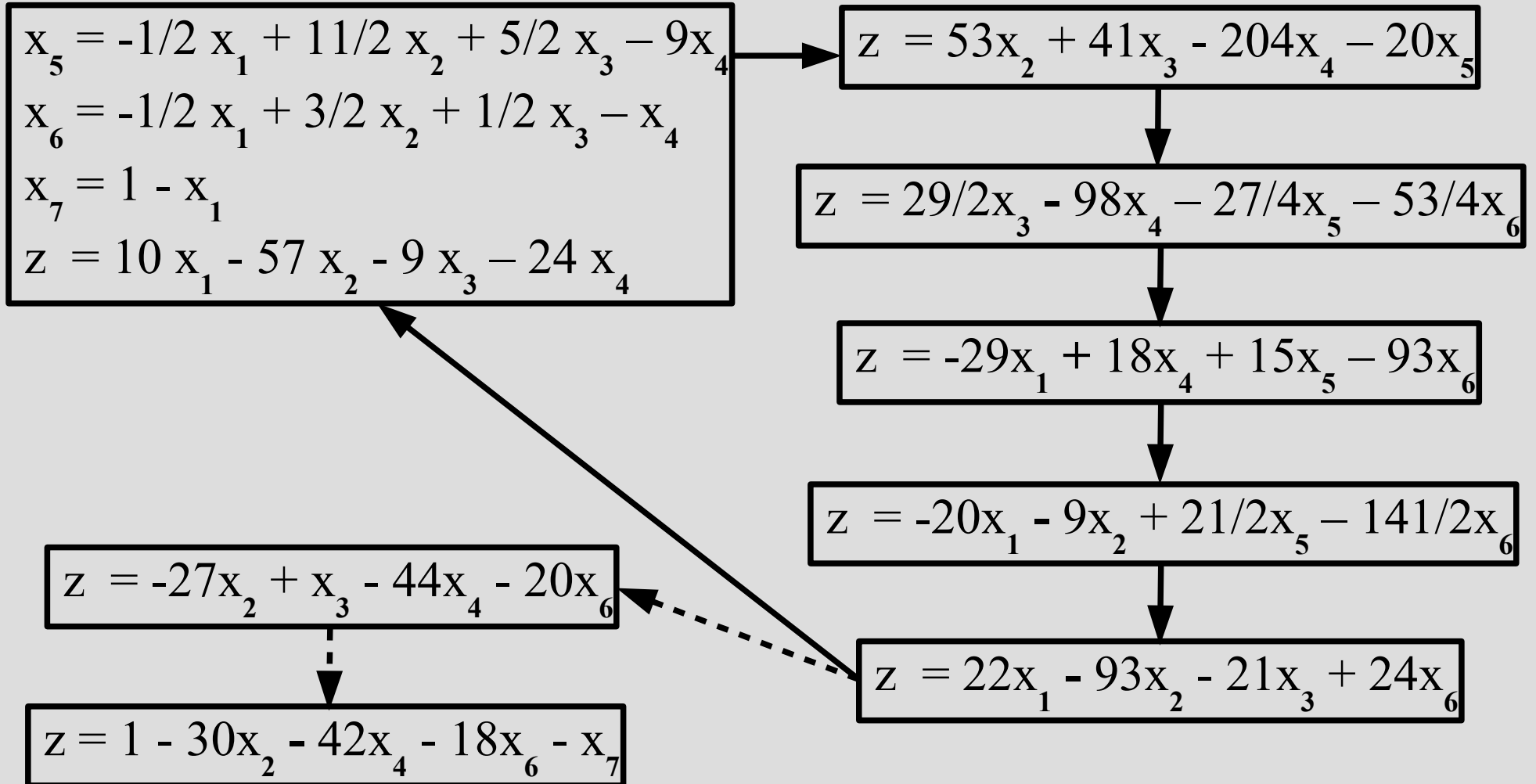
# Un exemple de cyclage (1/2)

- Pour décrire la version du simplexe utilisée, on a besoin de spécifier le choix de la variable :
  - Entrante : une dont le coût réduit est le plus grand
  - Sortante : s'il y a le choix, celle de plus petit indice

Dictionnaire obtenu pour ce PL après 6 itérations = dictionnaire initial (6 dictionnaires qui correspondent tous au même sommet, l'origine)

$$\left\{ \begin{array}{l} \max 10 x_1 - 57 x_2 - 9 x_3 - 24 x_4 \\ \text{sous contraintes :} \\ 0.5 x_1 - 5.5 x_2 - 2.5 x_3 + 9 x_4 \leq 0 \\ 0.5 x_1 - 1.5 x_2 - 0.5 x_3 + x_4 \leq 0 \\ x_1 \leq 1 \\ x_1 \geq 0, x_2 \geq 0, x_3 \geq 0, x_4 \geq 0 \end{array} \right.$$

# Un exemple de cyclage (2/2)



# Règle de Bland

- Des règles existent pour éviter le cyclage, dont la **règle de Bland**, décrivant le choix de la variable
  - Entrante : celle de coût réduit  $> 0$  de plus petit indice
  - Sortante : s'il y a le choix, celle de plus petit indice
    - Pour le PL précédent, on trouve alors une solution optimale (coûts réduits  $\leq 0$ ) de valeur  $1 > 0$ , après 7 itérations
  - En pratique, les dégénérescences et, plus encore, le cyclage, sont des phénomènes rares
    - S'il n'y en a pas, alors le simplexe converge souvent plus vite si la variable entrante est celle de plus grand coût réduit
    - Ainsi, on peut définir une stratégie mixte (on utilise cette règle ssi la solution associée au tableau courant est dégénérée)

# Dégénérescences et solutions optimales

- La présence de dégénérescences peut parfois gêner la reconnaissance de solutions optimales :

$$\begin{aligned}x_1 &= 2 + 2/7 x_3 - 1/7 x_6 \\x_2 &= 5 - 1/14 x_3 - 3/14 x_6 \\x_4 &= -5/7 x_3 + 6/7 x_6 \\x_5 &= 1 - 2/7 x_3 + 1/7 x_6 \\z &= 7 + 3/14 x_3 - 5/14 x_6\end{aligned}$$

Un dictionnaire  
(de valeur 7, donc optimal)

$$\begin{aligned}x_1 &= 2 - 2/5 x_4 + 1/5 x_6 \\x_2 &= 5 + 1/10 x_4 - 3/10 x_6 \\x_3 &= -7/5 x_4 + 6/5 x_6 \\x_5 &= 1 + 2/5 x_4 - 1/5 x_6 \\z &= 7 - \underbrace{3/10 x_4 - 1/10 x_6}_{\text{(Coûts réduits } < 0 \text{)}}\end{aligned}$$

Dictionnaire optimal  
(de valeur 7)

Le dictionnaire initial, bien qu'optimal, a un coût réduit  $> 0$  : c'est l'un des effets du phénomène de dégénérescence !

# Une seconde complication : et si l'origine ne compte pas ?

- Solution admissible initiale pour un PL P ?
  - Origine = solution initiale associée aux var. d'écart
  - Contrainte  $x_1 \geq 5 \Rightarrow$  origine non admissible (cas **défavorable**, par opposition au cas **favorable**)
    - La contrainte  $x_1 - e_1 = 5$  la rend non admissible
    - La contrainte  $x_1 - e_1 + a_1 = 5$  la rendrait admissible
    - Ces 2 contraintes sont équivalentes ssi  $a_1 = 0$
  - Idée pour trouver une solution admissible (**Phase 1**) :
    - Modifier les contraintes (ajouter les  $a_i$ )
    - Chercher à rendre les  $a_i$  nuls, en min. leur somme

# Phase 1 et PL auxiliaire

- On appelle **PL auxiliaire** le PL  $P'$  obtenu en
  - Partant de la FS de  $P$  (hypothèse non restrictive :  $2^{\text{nds}}$  membres positifs),
  - Ajoutant une variable **artificielle** (ou **auxiliaire**) sur chaque contrainte non vérifiée par l'origine,
  - Remplaçant la fonction de coût par la minimisation de la somme des variables auxiliaires.
- La **Phase 1** consiste à résoudre  $P'$ 
  - Si sa valeur optimale est nulle ( $a_i = 0$  pour tout  $i$ ), on a une solution admissible pour  $P$ ,
  - Sinon,  $P$  n'admet aucune solution admissible !

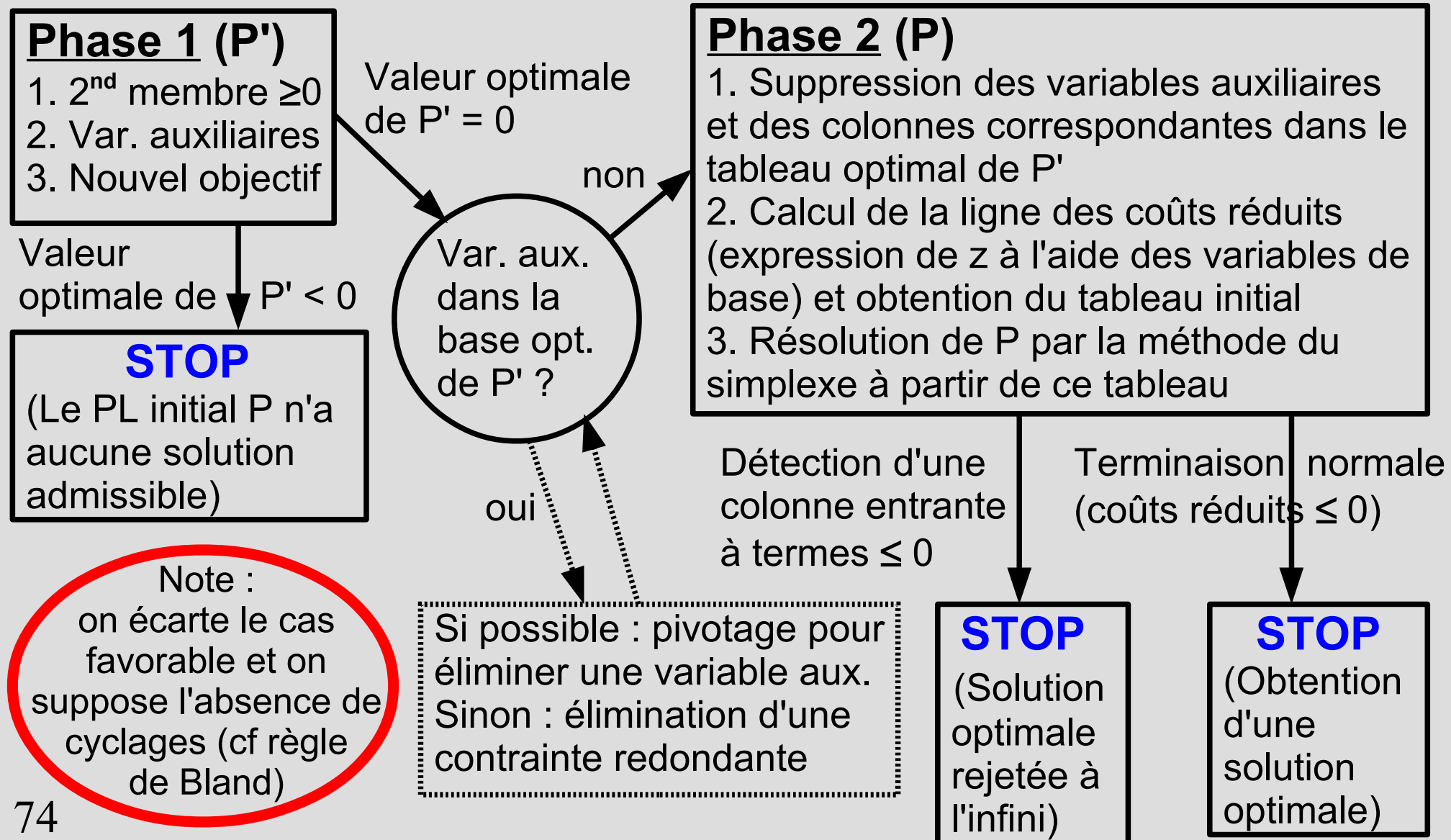
# Tableaux initiaux de P et P'

- Si  $x^*, a^*$  est une solution optimale non dégénérée de P' de valeur 0, alors :
    - $a^*=0$ , et toute variable auxiliaire est donc hors-base
    - **$x^*$  est donc une solution admissible pour P**, et le tableau initial du simplexe pour P est obtenu **en supprimant les colonnes associées aux var. aux.  $a^*$**
  - Tableau initial de P' ?
    - Base initiale : variable en base sur la  $j^e$  ligne = variable auxiliaire s'il y en a une, variable d'écart sinon
    - Solution initiale admissible car 2<sup>nd</sup> membre positif
- 72 – Il reste à réécrire la fonction objectif  $z' = \max \sum_i -a_i \dots$

# Tableau initial de P (cas général)

- Et si la solution optimale de P' est de valeur nulle, mais il reste des variables artificielles en base ?
  - Ces variables étant nulles, la solution est dégénérée !
  - On peut éliminer ces variables de la base :
    - Si la  $k^e$  variable de base est une variable artificielle, on choisit l'élément en colonne  $j$  de cette ligne
      - Non nul (et donc pas en base)
      - Correspondant à une variable du PL initial
    - Si  $j$  n'existe pas, alors la matrice  $M$  n'est pas de rang plein, et la ligne  $k$  correspond donc à une contrainte **redondante**
    - Sinon  $k$  sort de la base,  $j$  rentre dans la base, puis pivotage du tableau (tableau associé à la nouvelle base)

# Bilan : un algorithme COMPLET pour la résolution des PL



# A retenir...

- Méthode du simplexe
  - Solutions de base admissibles, dictionnaires et tableaux du simplexe
  - Variable entrante, variable sortante, coûts réduits
- Dégénérescences
  - Solution de base dégénérée, cyclage, règle de Bland
- Méthode des deux phases
  - Variables artificielles/auxiliaires, PL auxiliaire
  - Variante : méthode du grand  $M$  (la fonction objectif s'écrit alors  $\max c(x) - M \cdot \sum_i a_i$ )

# Programmation Linéaire

Troisième partie :

Dualité en  
Programmation Linéaire

# La dualité : point de vue économique (1/5)

- On considère à nouveau le PL du brasseur :

$$\max 15 x_1 + 25 x_2$$

sous contraintes :

$$2.5 x_1 + 7.5 x_2 \leq 240 \text{ (maïs)}$$

$$0.125 x_1 + 0.125 x_2 \leq 5 \text{ (houblon)}$$

$$17.5 x_1 + 10 x_2 \leq 595 \text{ (malt)}$$

$$x_1 \geq 0, x_2 \geq 0$$

# La dualité : point de vue économique (2/5)

- Rappel : ce PL est équivalent au PL suivant

$$\max 15 x_1 + 25 x_2$$

sous contraintes :

$$x_1 + 3 x_2 \leq 96 \text{ (maïs)}$$

$$x_1 + x_2 \leq 40 \text{ (houblon)}$$

$$7 x_1 + 4 x_2 \leq 238 \text{ (malt)}$$

$$x_1 \geq 0, x_2 \geq 0$$

# La dualité : point de vue économique (3/5)

- Pour produire sa bière, le brasseur doit auparavant acheter ses matières premières (MP)
  - maïs, houblon et malt
- A partir de quel(s) prix d'achat des MP l'activité de production de bière n'est-elle plus rentable ?
  - Coût de fabrication d'une UV  $\geq$  Prix de vente d'une UV
  - On cherche donc à minimiser la somme des prix des MP sous les 2 contraintes de non rentabilité
    - Soit  $u_i \geq 0$  le prix de la  $i^e$  MP : ce problème d'achat des MP peut se modéliser par un PL à 3 variables et 2 contraintes

# La dualité : point de vue économique (4/5)

- PL obtenu :

$$\min 240 u_1 + 5 u_2 + 595 u_3$$

sous contraintes :

$$2.5 u_1 + 0.125 u_2 + 17.5 u_3 \geq 15 \text{ (bière blonde)}$$

$$7.5 u_1 + 0.125 u_2 + 10 u_3 \geq 25 \text{ (bière brune)}$$

$$u_1 \geq 0, u_2 \geq 0, u_3 \geq 0$$

- Réécriture du PL ?

- Via un changement de variables... (Ex. :  $y_3 = 2.5 u_3$ )

# La dualité : point de vue économique (5/5)

- On obtient un nouveau PL (qui est le PL «**dual**» du PL «**primal**» = PL du brasseur modifié) :

$$\min 96 y_1 + 40 y_2 + 238 y_3$$

sous contraintes :

$$y_1 + y_2 + 7 y_3 \geq 15 \text{ (bière blonde)}$$

$$3 y_1 + y_2 + 4 y_3 \geq 25 \text{ (bière brune)}$$

$$y_1 \geq 0, y_2 \geq 0, y_3 \geq 0$$

# Bilan : écriture du PL dual

- Primal P :  $\max c(x)$ , avec  $x \in A$  et  $x \geq 0$
- Règles de passage Primal P  $\rightarrow$  Dual D :
  - « max »  $\rightarrow$  « min »
  - $b \rightarrow c$  ( $b = 2^{\text{nds}}$  membres de P)
  - $c \rightarrow b$
  - lignes  $\rightarrow$  colonnes
  - «  $\leq$  »  $\rightarrow$  «  $\geq$  »
  - m contraintes primales  $\rightarrow$  m variables duales
  - n variables primales  $\rightarrow$  n contraintes duales

◇ ? Propriété : la dualisation (passage primal  $\rightarrow$  dual) est une opération **involutive**. (Le dual du dual est le primal.)

# Dualiser un PL primal : s'affranchir de la FC ?

- Comment écrire le dual directement si on a :
  - Contrainte 1 (maïs) de la forme  $x_1 + 3 x_2 \geq 96 (y_1)$  ?
    - Forme équivalente :  $-x_1 - 3 x_2 \leq -96 (y'_1)$ , avec  $y'_1 \geq 0$
    - On a alors  $y_1 = -y'_1$ , et donc  $y_1 \leq 0$
  - Contrainte 1 (maïs) de la forme  $x_1 + 3 x_2 = 96 (y_1)$  ?
    - Forme équivalente :  $x_1 + 3 x_2 \leq 96 (y^+_1)$  et  $-x_1 - 3 x_2 \leq -96 (y^-_1)$ , avec  $y^+_1 \geq 0$  et  $y^-_1 \geq 0$
    - On a alors  $y_1 = y^+_1 - y^-_1$ , et donc  $y_1 \in \mathbb{R}$

# Bilan : écriture du PL dual à partir d'un PL primal quelconque (1/2)

PRIMAL	max	min	DUAL
contraintes	$\leq b_i$ $\geq b_i$ $= b_i$	$\geq 0$ $\leq 0$ réelle	variables
variables	$\geq 0$ $\leq 0$ réelle	$\geq c_j$ $\leq c_j$ $= c_j$	contraintes

# Bilan : écriture du PL dual à partir d'un PL primal quelconque (2/2)

DUAL	max	min	PRIMAL
contraintes	$\leq b_i$ $\geq b_i$ $= b_i$	$\geq 0$ $\leq 0$ réelle	variables
variables	$\geq 0$ $\leq 0$ réelle	$\geq c_j$ $\leq c_j$ $= c_j$	contraintes

# Théorèmes de dualité

- Intérêt du dual ? Lien entre primal P et dual D ?
- Théorème de dualité **faible (TDFa)**
  - **La valeur de toute solution admissible de P est au plus celle de toute solution admissible de D**
    - Preuve simple (x et y admissibles pour P et D resp.) :  
$$c(x) = \sum_j c_j x_j \leq \sum_j (\sum_i m_{ij} y_i) x_j = \sum_i (\sum_j m_{ij} x_j) y_i \leq \sum_i y_i b_i = \sum_i b_i y_i$$
- Théorème de dualité **forte (TDFo)**
  - **Si P admet une solution optimale  $x^*$ , alors P et D ont la même valeur optimale**
    - Preuve (idée) :  $x^*$  a même valeur que  $y_i^*$  = -coût réduit ( $\leq 0$ ) de la  $i^e$  var. d'écart de P dans le tableau de  $x^*$

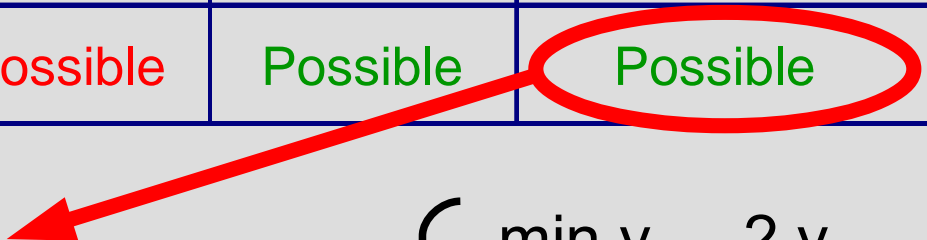
# TDFa et TDFo : bilan

		Primal		
		Optimum fini	Non borné	Non admissible
Dual	Optimum fini	Possible	Impossible	Impossible
	Non borné	Impossible	Impossible	Possible
	Non admissible	Impossible	Possible	Possible

$$\left\{ \begin{array}{l} \max 2x_1 - x_2 \\ x_1 - x_2 \leq 1 \\ -x_1 + x_2 \leq -2 \\ x_1 \geq 0, x_2 \geq 0 \end{array} \right.$$



$$\left\{ \begin{array}{l} \min y_1 - 2y_2 \\ y_1 - y_2 \geq 2 \\ -y_1 + y_2 \geq -1 \\ y_1 \geq 0, y_2 \geq 0 \end{array} \right.$$



# Utilisation du TDFo

- Observation : on peut résoudre le primal à l'aide du dual... Mais quand a-t-on intérêt à le faire ?
  - 1<sup>er</sup> cas : le primal P a seulement 2 contraintes (et le dual D a donc 2 variables), mais plus de 4 variables
    - On peut alors résoudre D graphiquement, mais pas P !
  - 2<sup>e</sup> cas : le primal P a peu de variables, mais un nombre élevé de contraintes
    - Le dual a alors peu de contraintes, et peut donc être efficacement résolu par le simplexe !

# Conséquences du TDFO

- Test d'optimalité pour un PL P
  - Soit  $x$  solution admissible pour P et  $y$  solution admissible pour le dual D, alors  $x$  est optimale pour P et  $y$  est optimale pour D ssi elles ont même valeur
- Pour un PL P et son dual D, équivalence entre
  - Condition d'optimalité de P (coûts réduits  $\leq 0$ )
  - Condition d'admissibilité de D (variables duales  $\geq 0$ )
- Théorème des écarts complémentaires (**TEC**)

# Théorème des écarts complémentaires (TEC)

- **TEC** : deux solutions  $x^*$  (admissible pour un PL donné  $P$ ) et  $y^*$  (admissible pour le dual  $D$  de  $P$ ) sont optimales pour  $P$  et  $D$  respectivement ssi
    - (1) Pour tout  $j$ , la  $j^{\text{e}}$  variable de  $x^*$  est nulle **OU** la  $j^{\text{e}}$  contrainte duale est saturée (vérifiée à l'égalité)
- ET**
- (2) Pour tout  $i$ , la  $i^{\text{e}}$  variable de  $y^*$  est nulle **OU** la  $i^{\text{e}}$  contrainte primale est saturée (vérifiée à l'égalité)

# Preuve du TEC

- Démonstration (avec  $P : \{\max c(x), x \in A, x \geq 0\}$ ) :
  - On a  $c_j x_j^* \leq (\sum_i m_{ij} y_i^*) x_j^*$  et  $(\sum_j m_{ij} x_j^*) y_i^* \leq b_i y_i^*$   
(b= 2nds membres de P)
  - Ces deux inégalités impliquent :
$$\sum_j c_j x_j^* \leq \sum_j (\sum_i m_{ij} y_i^*) x_j^* = \sum_i (\sum_j m_{ij} x_j^*) y_i^* \leq \sum_i b_i y_i^*$$
(On a l'égalité ssi tous les  $\leq$  sont des =)
  - On a donc  $\sum_j c_j x_j^* = \sum_i b_i y_i^*$  si et seulement si  $\sum_j (c_j - \sum_i m_{ij} y_i^*) x_j^* = 0$  et  $\sum_i (b_i - \sum_j m_{ij} x_j^*) y_i^* = 0$
  - D'après le TDFo,  $\sum_j c_j x_j^* = \sum_i b_i y_i^*$  ssi  $x^*$  et  $y^*$  sont simultanément optimales

# Utilisation pratique du TEC

- **En pratique, une solution admissible  $x^*$  de P est optimale ssi la solution  $y^*$ , calculée de façon à satisfaire (1)+(2) avec  $x^*$ , est admissible pour le dual D de P**
  - En effet, si  $x^*$  est optimale pour P, alors :
    - D a une solution optimale (donc admissible)  $y^*$  (cf TDFo)
    - Ces deux solutions optimales satisfont (1)+(2) (cf TEC)
  - Réciproquement, si  $y^*$  existe, elle est admissible pour le PL dual D, et donc, d'après le TEC, optimale

# Test d'optimalité utilisant le TEC

- Tester l'optimalité d'une solution  $x$  de  $P$  ?
  - On injecte la valeur des  $x_j$  dans (1)+(2)
  - On en déduit les valeurs des  $y_i$  via ces règles :
    - $x_j > 0 \Rightarrow j^{\text{e}}$  contrainte duale saturée
    - $i^{\text{e}}$  contrainte primale non saturée  $\Rightarrow y_i = 0$
  - Finalement, on vérifie si  $y$ , la solution duale ainsi obtenue, est admissible ou non pour le dual  $D$

# Exemple d'application du TEC

- On considère à nouveau le PL du brasseur :
  - Soit la solution primale  $x_1 = 0$  et  $x_2 = 32$ 
    - Solution duale associée :  $y_2 = y_3 = 0$ , et donc  $y_1 = 25/3$
    - Solution duale non admissible ( $y_1 < 15$ ), et donc solution primale considérée **non optimale** !
  - Soit à présent la solution primale  $x_1 = 12$  et  $x_2 = 28$ 
    - Solution duale associée :  $y_3 = 0$ , et donc  $y_1 = 5$  et  $y_2 = 10$  (solution du système  $\{y_1 + y_2 = 15, 3y_1 + y_2 = 25\}$ )
    - Solution duale admissible, et donc, d'après le TEC, solution primale considérée **optimale**

# Interprétation économique des variables duales (1/2)

- On a vu que les variables duales pouvaient être vues comme les prix maximaux des MP garantissant la rentabilité de la vente de bière
- Il existe une autre interprétation économique :
  - On appelle **coût marginal** de la  $i^{\text{e}}$  MP (la ressource associée à la  $i^{\text{e}}$  contrainte primale) la valeur dans la solution optimale de la  $i^{\text{e}}$  variable duale
    - Cf ex. précédent :  $y_1=5$  (maïs),  $y_2=10$  (houblon),  $y_3=0$  (malt)
  - Ce nom signifie qu'augmenter cette ressource de  $\varepsilon$  unités génère un profit égal à  $\varepsilon$  fois ce coût marginal (vrai pour  $\varepsilon$  suffisamment petit !) : ici  $\varepsilon_1=24$ ,  $\varepsilon_2=84/17$

# Interprétation économique des variables duales (2/2)

- Preuve (idée) ?
  - Utiliser la caractérisation matricielle des bases
  - Soit  $B$  une base optimale (non dégénérée) pour le PL primal  $P$ , et  $y^*$  la solution optimale du dual  $D$ 
    - Comme  $B^{-1}b > 0$ ,  $B$  reste admissible pour  $P$  si on remplace le 2<sup>nd</sup> membre  $b$  par  $(b + \Delta b)$  (où  $\Delta b = (0, \dots, 0, \varepsilon, 0, \dots, 0)$ )
    - $y^*$  indépendant de  $b$  (car coûts réduits  $= c^T - c_B^T B^{-1} M$ )
  - On a alors 2 solutions admissibles pour  $P$  et  $D$  vérifiant les conditions du TEC :  $y^*$  reste donc optimale !
    - Sa valeur est  $(b + \Delta b)^T y^*$ , d'où une variation de coût de  $\Delta b^T y^*$
    - Bilan : le critère à vérifier est donc  $B^{-1}(b + \Delta b) = B^{-1}b + B^{-1}\Delta b \geq 0$

# Analyse de sensibilité

- Soit  $x^*$  une solution optimale pour un PL  $P$  :
  - Si la fonction objectif  $c$  est légèrement modifiée, que devient la valeur optimale de  $P$  ? En particulier, pour quelles modifications (de  $c$ )  $x^*$  reste-t-elle optimale ?
  - On repart du tableau du simplexe associé à  $x^*$ 
    - Les variables de base sont écrites en fonction des autres
    - Si  $z$  change, le nouveau tableau associé à  $x^*$  est obtenu en écrivant le nouveau  $z$  à l'aide des variables hors-base
  - $x^*$  reste optimale ssi la solution duale associée reste admissible, et donc ssi tous les coûts réduits sont  $\leq 0$

# Analyse de sensibilité pour le problème du brasseur

## Dictionnaire optimal $x^*$

$$x_1 = 12 + 1/2 x_3 - 3/2 x_4$$

$$x_2 = 28 - 1/2 x_3 + 1/2 x_4$$

$$x_5 = 42 - 3/2 x_3 + 17/2 x_4$$

$$z = 15x_1 + 25x_2 \rightsquigarrow z = (15 + \delta)x_1 + 25x_2 ?$$

$$\hookrightarrow z = 880 + 12\delta + (-5 + \delta/2)x_3 + (-10 - 3\delta/2)x_4$$

Ainsi : coûts réduits  $\leq 0$  ssi  $(-5 + \delta/2) \leq 0$  ET  $(-10 - 3\delta/2) \leq 0$

**$x^*$  reste donc optimale ssi le coefficient de  $x_1$  dans  $z$  est compris entre  $15 - 20/3$  et  $25$**

# A retenir...

- Ecriture du PL dual, involutivité de la dualisation
- Théorèmes de dualité (TDFa, TDFo, TEC), et leurs conséquences...
- Interprétation économique des variables duales (coûts marginaux)
- Analyse de sensibilité

# Programmation Linéaire en Nombres Entiers

Première partie :

Le modèle de la  
Programmation Linéaire  
en Nombres Entiers

# Un exemple (1/2)

- Une entreprise fabrique deux sortes de produits, P1 et P2 :
  - Chaque unité de P1 nécessite 2 unités de MP, chaque unité de P2 en nécessite 4
  - La vente d'une unité de P1 rapporte 100 euros, la vente d'une unité de P2 rapporte 500 euros
  - 5 unités de MP sont disponibles
- On cherche le plan de production qui maximise le revenu de l'entreprise

## Un exemple (2/2)

- Solution optimale (méthode graphique) :
  - $x_1=0, x_2=1,25$  (de valeur 625)
- Impossible : on ne peut pas fabriquer  $\frac{1}{4}$  d'unité de P2 !!!
  - On veut donc que les deux variables  $x_1$  et  $x_2$  prennent des valeurs entières !
- Solution optimale (par énumération) :
  - $x_1=0, x_2=1$  (de valeur 500)

# En résumé...

- Pour certains problèmes, exiger des variables entières peut donc avoir un sens !
  - Parfois, cela implique plusieurs modèles possibles...
    - Ex. :  $\{x_1 + x_2 \leq 1, x_1 \in \mathbf{IN}, x_2 \in \mathbf{IN}\} = \{2x_1 + 4x_2 \leq 5, x_1 \in \{0, 1\}, x_2 \in \mathbf{IN}\}$
- Pour certains problèmes, la solution optimale en variables entières est différente de la solution optimale en variables réelles...
  - En particulier, le simplexe ne nous aide pas !
- PLNE = PL avec variables entières
  - PNE = PM avec variables entières

# Quelques familles de PNE

- Si les variables entières sont dans  $\{0,1\}$ 
  - Variables (bivalentes) 0-1, booléennes, binaires...
  - Equivalence avec des contraintes quadratiques
- Si certaines variables sont réelles/continues = P(L) en variables mixtes (P(L)VM)
  - Un PL est donc un PLVM particulier !
- Existe-t-il un lien entre PNE et PLNE (= une transformation pour passer de l'un à l'autre) ?

# Linéarisation(s)

- Linéariser = transformer un PNE en PLNE
- Peut-on linéariser un produit  $xy$  ?
  - Si  $x$  et  $y$  quelconques : en général, NON !
  - Si  $x, y \in \{0, 1\}$  : OUI, par ajout de contraintes !
    - En utilisant uniquement des variables 0-1
    - Avec des variables mixtes
- Plus généralement, comment linéariser un produit  $x_1 x_2 \dots x_k$ , avec  $x_i \in \{0, 1\}$  pour tout  $i$  ?

# Modélisation de pb classiques d'optimisation par la PLNE

- Problème du sac-à-dos
- Problème d'affectation linéaire
- Problèmes de graphe :
  - Couverture par les sommets
  - Clique
  - PVC

# Problème du sac-à-dos

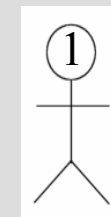
- **Contexte** : dans une bijouterie, un voleur équipé d'un simple sac-à-dos veut décider quels objets il doit subtiliser pour maximiser le coût total de revente de son butin
- **Formalisation** : chaque objet est muni d'un poids  $p_i$  et d'un coût de revente  $c_i$ , et le poids maximum que le sac peut supporter est  $B$
- **Modèle PLNE** ?
  - Une variable 0-1 par objet



# Problème d'affectation linéaire

- **Contexte** : dans une entreprise,  $n$  tâches doivent être affectées à  $n$  agents de façon à minimiser le temps total (= somme des temps) mis pour effectuer toutes les tâches, sachant que le temps mis par un agent pour effectuer une tâche dépend de la tâche ET de l'agent
- **Formalisation** : chaque paire (tâche, agent) est munie d'une valeur (temps en minutes), et 1 tâche = 1 agent
- **Modèle PLNE ?**
  - Une variable 0-1 par paire (tâche, agent)

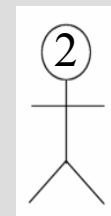
Tâche 1 Tâche 2 Tâche 3



10 min

15 min

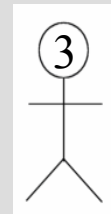
40 min



5 min

20 min

1 heure



10 min

3 heures

3 heures

Agents

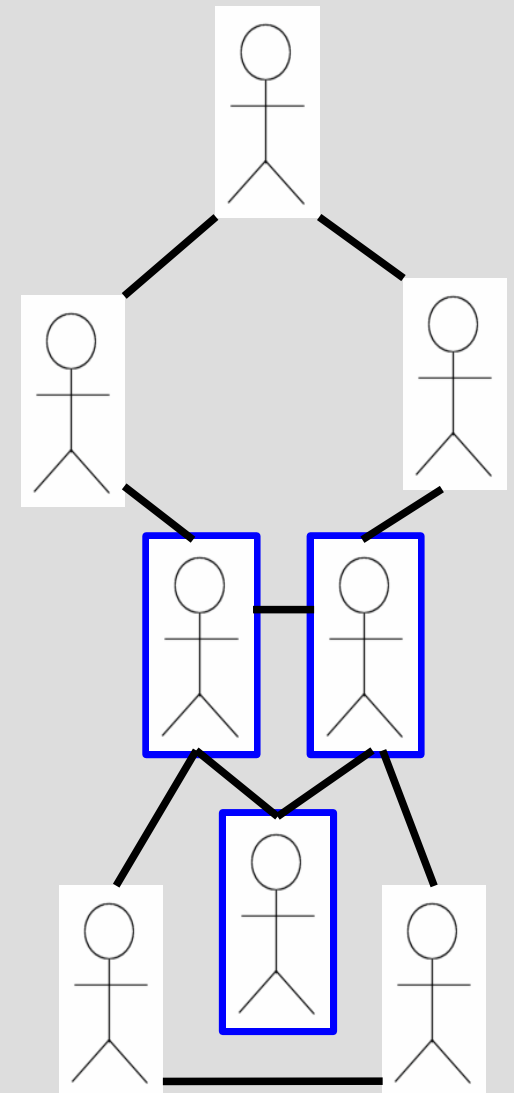
# Problème de la couverture minimum par les sommets

- **Contexte** : on veut placer le moins possible de caméras de surveillance à des carrefours de façon à surveiller toutes les rues d'un quartier
- **Formalisation** : on représente le quartier par un graphe, dont les sommets sont les carrefours et les arêtes les rues entre ces carrefours (on cherche alors un ensemble de sommets de taille minimum qui « couvre » toutes les arêtes du graphe)
- **Modèle PLNE ?**
  - Une variable 0-1 par sommet du graphe



# Problème de la clique maximum

- Dans un graphe  $G$ , une **clique** est un ensemble de sommets  $C$  de  $G$  tel que, pour toute paire  $x, y$  de sommets de  $C$ , il existe une arête de  $G$  entre  $x$  et  $y$ 
  - **Application** : communautés (cliques) de grande taille dans les réseaux sociaux, où 2 personnes sont reliées si elles se connaissent (par exemple)
- Comment modéliser le pb de recherche d'une clique de taille maximum par la PLNE ?
  - Une variable 0-1 par sommet de  $G$
  - Modèle PNE ?
  - Modèle PLNE par linéarisation ?



# Problème du voyageur de commerce - PVC (1/2)

- **Contexte** : un représentant de commerce (allemand) doit effectuer sa tournée de  $n$  villes, consistant à visiter chacune des villes une et une seule fois
- **Objectif** : minimiser la distance totale parcourue
- **Modèle** (graphe) : graphe  $G=(\{\text{villes}\},A)$ , dont les arêtes  $A$  sont pondérées par des distances ; on cherche alors dans  $G$  un cycle hamiltonien (CH) de longueur minimum



# Problème du voyageur de commerce - PVC (2/2)

- Modèle PLNE ?
  - Une variable 0-1 par arête de  $G$
  - Minimiser la longueur totale du CH, c'est-à-dire la somme des longueurs des arêtes du CH
  - Deux types de contraintes :
    1. Tout sommet est adjacent à deux arêtes du CH
    2. Elimination des sous-tours ?

Pour tout sous-ensemble non vide de villes, il y a au moins une arête ayant une extrémité « à l'extérieur »
  - Nombre de contraintes à ajouter ?

# Modéliser une alternative

- Soit le problème d'optimisation suivant :
  - $\min\{c(x) \text{ avec } x \in A, 0 \leq x_1 \leq 10 \text{ et } (x_1 \leq 3 \text{ OU } x_1 \geq 7)\}$ 
    - Par exemple,  $x_1 = 5 \Rightarrow$  non admissible
  - Disjonction de contraintes : **NON LINEAIRE !**
  - Comment le modéliser par un PLVM ?
- Idée : introduire une variable 0-1  $y$  telle que
  - $x_1 \leq 3$  si  $y=0$  (et  $x_1 \geq 7$  est alors « inactive »)
  - $x_1 \geq 7$  si  $y=1$  (et  $x_1 \leq 3$  est alors « inactive »)

# Un bref mot sur la résolution des P(L)NE (1/3)

- Résoudre le PL associé et arrondir la valeur des variables continues de la solution ? NON !
- 1er contre-exemple :

$$\max\{x+Ky, \text{ s.c. } 2x+4y \leq 3, x \geq 0, y \geq 0\}$$

- 3 sommets :  $(0,0)$ ,  $(3/2,0)$ ,  $(0, 3/4)$
- Solution continue optimale :  $y=3/4$ , de valeur  $3K/4$
- Solution entière optimale (énumération sur les 2 solutions  $(0,0)$  et  $(1,0)$ ) :  $x=1$ , de valeur 1

# Un bref mot sur la résolution des P(L)NE (2/3)

- 2e contre-exemple :

max  $4x+5y$ , sous contraintes :

$$x+y \leq 4.5$$

$$2x+y \geq 5.5$$

$$x \in [0,3], y \geq 1/2$$

- 4 sommets :  $(1, 7/2)$ ,  $(3, 1/2)$ ,  $(3, 3/2)$ ,  $(5/2, 1/2)$
- Solution continue optimale :  $x=1$ ,  $y=7/2$  de valeur 21.5
- Solution entière optimale (énumération sur les 2 solutions  $(3,1)$  et  $(2,2)$ ) :  $x=2$ ,  $y=2$  de valeur 18

# Un bref mot sur la résolution des P(L)NE (3/3)

- Comment résoudre un PLNE ?
  - On ne peut utiliser aucun algorithme de PL (simplexe, points intérieurs, ellipsoïdes, ...)
  - Enumérer les sol. admissibles (ensemble discret) ?
    - Pour un PLNE avec 100 variables 0-1 :  $2^{100} \approx 10^{30}$  sol. potentielles à énumérer (**explosion combinatoire**) !
  - En fait, aucun algorithme efficace connu !
- Plusieurs approches d'efficacité variable, complémentaires et imparfaites, peuvent être utilisées (compromis temps/optimalité)

# A retenir...

- Linéarisations
- Problèmes classiques et PLNE
- Apport des variables entières : modéliser des alternatives à l'aide de variables entières
- Explosion combinatoire

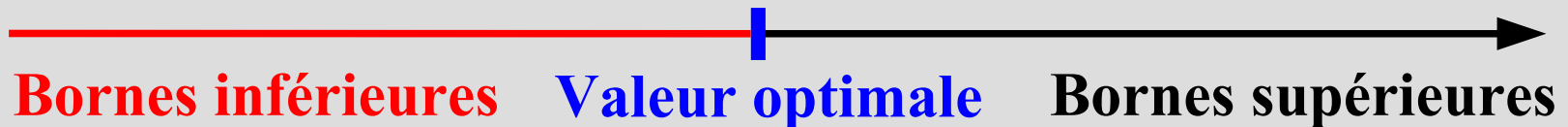
# Programmation Linéaire en Nombres Entiers

Deuxième partie :

Résolution (exacte) de  
Programmes Linéaires  
en Nombres Entiers

# Bornes et relaxations

- Comment résoudre de façon exacte (= trouver une solution optimale à) un PLNE  $P = \{\max c(x) \text{ avec } x \in A \text{ et } x \text{ entier}\}$  ?
  - Obtenir une **évaluation** (**borne**) de la valeur optimale
  - Déterminer une borne inférieure de cette valeur ?
    - Valeur de n'importe quelle solution admissible !
  - Déterminer une borne supérieure de cette valeur ?
    - En général, on a recours à des **relaxations**
  - Si ces deux bornes sont égales, alors on a fini !



# Bornes primales

- Borne **primale** = valeur d'une solution admissible
  - Si on maximise, c'est une borne inférieure
  - Si on minimise, c'est une borne supérieure
- Parfois, trouver une borne primale est simple :
  - Problème du sac-à-dos
- Parfois, ça l'est moins :
  - PVC dans un graphe quelconque
- Mais trouver une **bonne** borne primale est souvent une tâche difficile...

# Bornes duales

- Borne **duale**
  - Si on maximise, c'est une borne supérieure
  - Si on minimise, c'est une borne inférieure
- Déterminer des bornes duales via des relaxations
  - Idéalement, problème plus simple (plus rapide) à résoudre que le problème initial
  - Problème dont chaque solution a une valeur plus grande (si on maximise) ou plus petite (sinon) que celle de toute solution admissible du problème initial

# Définition et propriétés des relaxations

- Définition limitée à dessein au cadre de ce cours :  
Un problème (R) est une **relaxation** d'un PLNE (P) **si toute solution de (P) est solution de (R)**



1. **Si (R) est une relaxation de (P), alors sa valeur optimale est une borne duale pour (P)**
2. Si (R) n'a pas de solutions admissibles, (P) non plus
3. Si une solution optimale de (R) est admissible pour (P), alors elle est également optimale pour (P)

# Relaxation continue (RC)

- Soit le PLNE  $(P) = \{\max c(x) \text{ avec } x \in A, x \text{ entier}\}$ 
  - Question : relaxation  $(R)$  plus simple à résoudre ?
- Le PL  $(R) = \{\max c(x) \text{ avec } x \in A\}$  est appelé **relaxation continue** (ou linéaire) de  $(P)$ 
  - Dans  $(R)$ , on ignore les contraintes d'intégrité de  $(P)$
  - Propriété :  $(R)$  est bien une relaxation de  $(P)$ , car toute solution de  $(P)$  est solution de  $(R)$  !
  - Corollaire (cf brasseur) : si  $(R)$  admet une solution optimale entière, cette solution est optimale pour  $(P)$  !
  - Intérêt de  $(R)$  :  $(R)$  est un PL, et peut donc être résolu à l'aide d'algorithmes efficaces (simplexe) !

# Exemple de RC

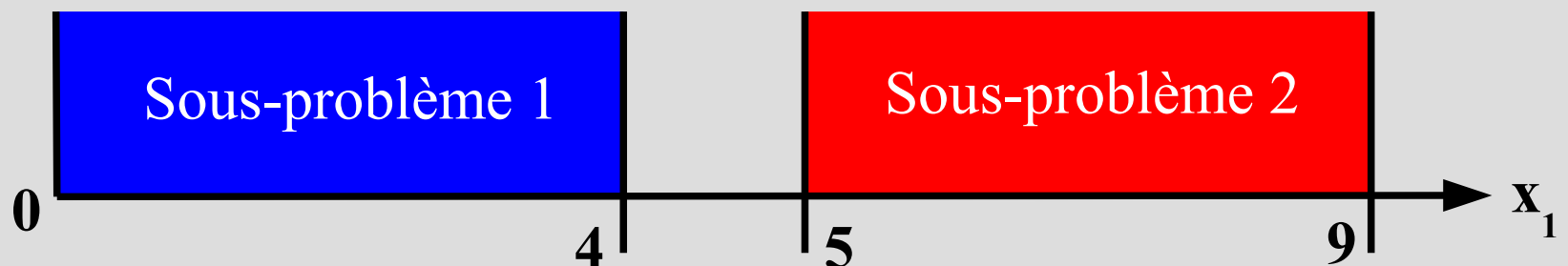
- Soit  $(P) = \{\max x_1 + 3x_2 \text{ s.c. } 2x_1 + 4x_2 \leq 5 \text{ et } (x_1, x_2) \in \mathbb{IN}^2\}$
- RC :  $(R) = \{\max x_1 + 3x_2 \text{ s.c. } 2x_1 + 4x_2 \leq 5, x_1 \geq 0, x_2 \geq 0\}$ 
  - Solution optimale de  $(R)$  :  $x_1 = 0, x_2 = 1.25$  de valeur 3.75
  - Ainsi, la solution entière  $(x_1 = 0, x_2 = 1)$  de valeur 3 est optimale pour  $(P)$ , car  $x_1 + 3x_2 \leq 3.75$ , et donc  $x_1 + 3x_2 \leq 3$
  - Le PLNE  $\{\max x_1 + 3x_2 \text{ s.c. } 4x_1 + 8x_2 \leq 11 \text{ et } (x_1, x_2) \in \mathbb{IN}^2\}$  a le même ensemble admissible que  $(P)$ , mais la valeur optimale de sa RC est  $33/8$  : la qualité de cette borne duale dépend donc du PLNE modélisant le problème !

# Séparation et évaluation (1/7)

- En anglais : Branch & Bound (CPLEX)
- Méthode (exacte) pour résoudre des P(L)NE, de la famille des méthodes de recherche arborescente
  - Idée : résoudre le P(L)NE en faisant une énumération intelligente (non exhaustive) des solutions admissibles
    - Enumération = séparation (branchement)
  - Objectif : ne pas développer tout l'arbre de recherche (AR), en « coupant » des branches à l'aide de bornes
    - Générer des bornes primales (solutions admissibles) ?
    - Générer des bornes duales (évaluation) via une relaxation ?
      - Utiliser la relaxation continue !

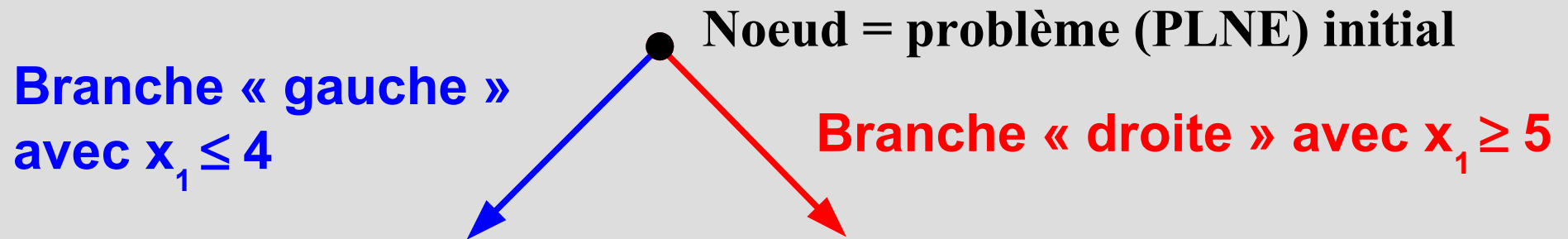
# Séparation et évaluation (2/7)

- Etape de séparation = diviser le pb en plusieurs sous-pb, de façon à ce que la résolution de tous ces sous-pb garantisse la résolution du pb initial
  - En pratique, on effectue un « branchement » dans l'AR, tout en garantissant la conservation de toutes les solutions admissibles du pb avant branchement
  - Par exemple, un PLNE ayant une variable  $x_1 \in \{0,9\}$  peut être résolu en posant d'abord  $x_1 \leq 4$ , puis  $x_1 \geq 5$



# Séparation et évaluation (3/7)

- Représentation ?



- Exemple d'AR (branchement uniquement) :
  - Soit le PLNE :  $\{\max 4x_1 + 8x_2 + 5x_3 \text{ avec } 2x_1 + 3x_2 + 2x_3 \leq 4 \text{ et } x_1, x_2, x_3 \in \{0, 1\}\}$
  - AR à 15 noeuds (8 feuilles), ou 13 noeuds (7 feuilles)

# Séparation et évaluation (4/7)

- Evaluer un noeud = calculer une « estimation » (borne duale) de la valeur optimale de ce PLNE
  - On résout la RC de ce PLNE (PLNE initial + étapes)
  - Racine = PLNE initial (pas de contraintes en plus)
- Intérêt de l'évaluation :
  - La borne duale calculée avec l'hypothèse  $x_1 \leq 4$  est différente de celle calculée avec l'hypothèse  $x_1 \geq 5$
  - Ces bornes peuvent permettre d'élaguer l'AR
  - Pour cela, une borne primale peut être nécessaire...

# Séparation et évaluation (5/7)

- Procédure : on évalue un noeud, puis on sépare
  - Sur la valeur d'une variable fractionnaire (non entière) dans la solution optimale de la RC
  - Exemple : si  $x_1=4.3$ , on pose d'abord  $x_1 \leq 4$ , puis  $x_1 \geq 5$
- Parfois, on peut « couper » une branche de l'AR :
  1. Si la valeur optimale de la RC est moins bonne que la borne primale courante (noeud sans intérêt)
  2. Si la RC n'admet aucune solution admissible
  3. Si la RC admet une solution entière

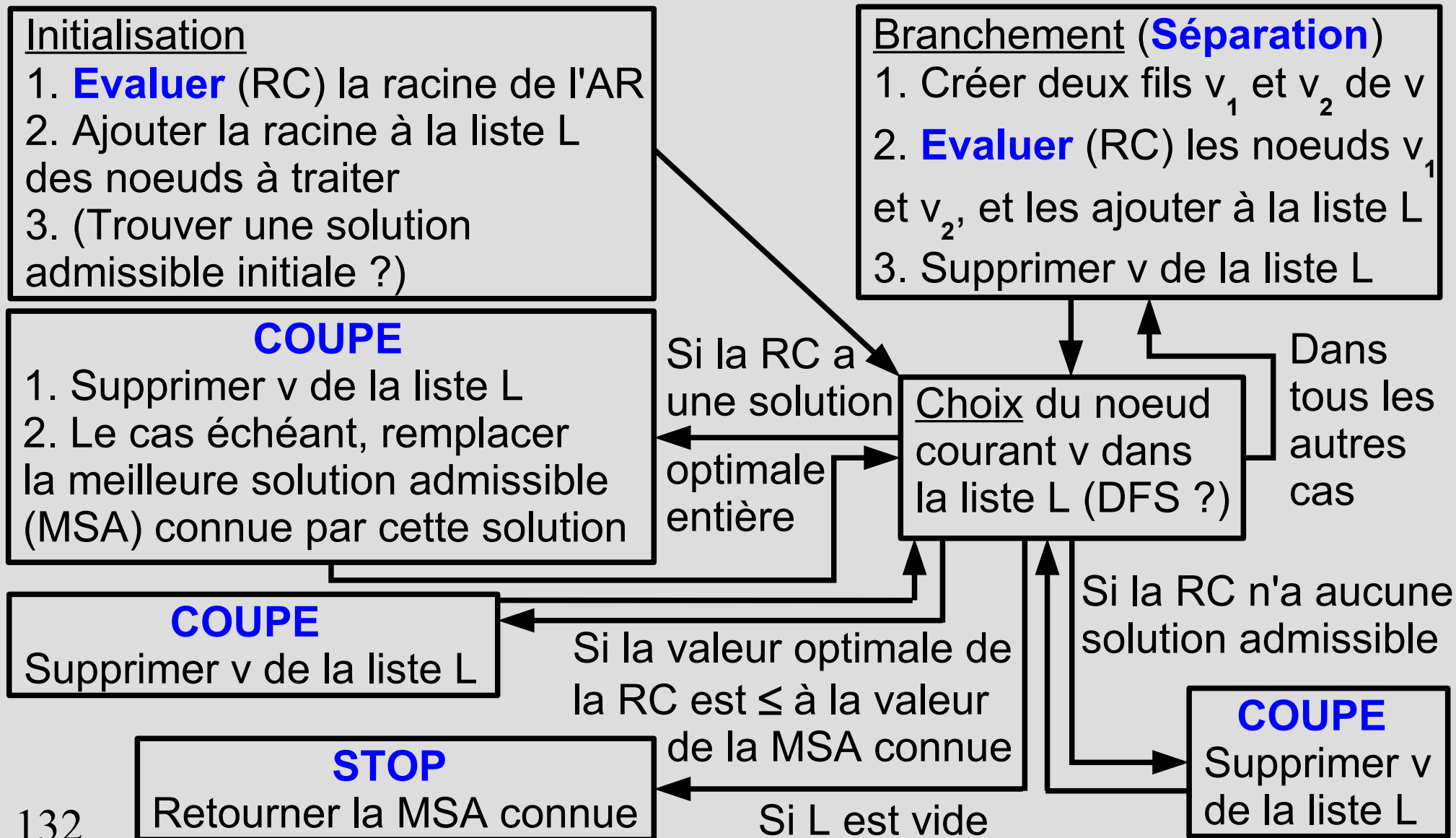
# Séparation et évaluation (6/7)

- Couper suppose l'existence d'une borne primale !
  - Meilleure est cette borne, plus on risque de couper
  - Parfois, lors de la construction de l'AR, l'évaluation du noeud fournit une solution entière (borne primale) : si nécessaire, actualiser alors la borne primale courante
  - On peut aussi essayer de générer par différents biais une borne primale (solution admissible) initiale...

# Séparation et évaluation (7/7)

- Réglages/choix du concepteur :
  - Déjà évoqués
    - Choix d'une borne primale, d'une borne duale
  - Sur quelle variable effectuer la séparation ?
    - Variable fractionnaire
      - Plus grande partie fractionnaire ?
      - Autre stratégie ?
  - Quelle stratégie adopter pour explorer l'AR ?
    - Profondeur d'abord (DFS)
    - Noeuds de coupe d'abord, puis noeud ayant la meilleure évaluation parmi les noeuds encore non explorés

# Bilan : séparation et évaluation (version maximisation)



# Exemple 1 : le sac-à-dos (1/2)

- Pourquoi ce choix ?
  - Variables 0-1
    - Hypothèses de branchement simples à faire/à intégrer
  - Calcul de la borne duale simple
    - Solution optimale de la RC très rapide à calculer
- Soit un SAD :  $\{\max c(x), a_1x_1 + \dots + a_nx_n \leq b, x \in \{0, 1\}^n\}$ 
  - On peut supposer que  $c_1/a_1 \geq c_2/a_2 \geq \dots \geq c_n/a_n$
  - Sol. opt. de la RC : on met à 1 toutes les variables  $x_1, x_2, \dots, x_{i-1}$  tant que possible, puis on prend la plus grande valeur  $< 1$  possible pour  $x_i$  (Exercice)

# Exemple 1 : le sac-à-dos (2/2)

- Instance considérée :

$$\max 15x_1 + 18x_2 + 4x_3 + 2x_4 + 5x_5 + x_6$$

$$3x_1 + 4x_2 + x_3 + x_4 + 3x_5 + x_6 \leq 5$$

$$x_i \in \{0,1\} \text{ pour tout } i \text{ entre } 1 \text{ et } 6$$

- On a bien :  $15/3 > 18/4 > 4/1 > 2/1 > 5/3 > 1/1$ 
  - Borne primale (solution) initiale :  $x_i = 0$  pour tout  $i$
  - Borne duale initiale :  $15 + 9 = 24$
- L'AR contient 5 noeuds

# Exemple 2 : PLNE à 2 variables

- Soit le PLNE suivant :

$$\min 4x - 5y$$

$$x + 1.5y \leq 4.5$$

$$2x + y \geq 5.5$$

$$x \in [0, 3], y \geq 1/2, x \text{ et } y \text{ entiers}$$

- Solution initiale :  $x=1.875$  et  $y=1.75$
- En fonction du choix (séparer sur  $x$  ou  $y$ ) : l'AR a 5 ou 7 noeuds !

# A retenir...

- La méthode par S&E :
  - Fournit une solution optimale en évitant autant que possible l'énumération explicite de toutes les sol. admissibles (énumération implicite)
  - Basée sur l'utilisation de bornes primales/duales
  - Inconvénient : parfois gourmande en temps, et en mémoire (ajout de contraintes en chaque noeud de l'AR), notamment si les bornes primales et/ou duales ne sont pas suffisamment bonnes
  - Peut donc nécessiter des alternatives...
    - Lesquelles ?

# Programmation Linéaire en Nombres Entiers

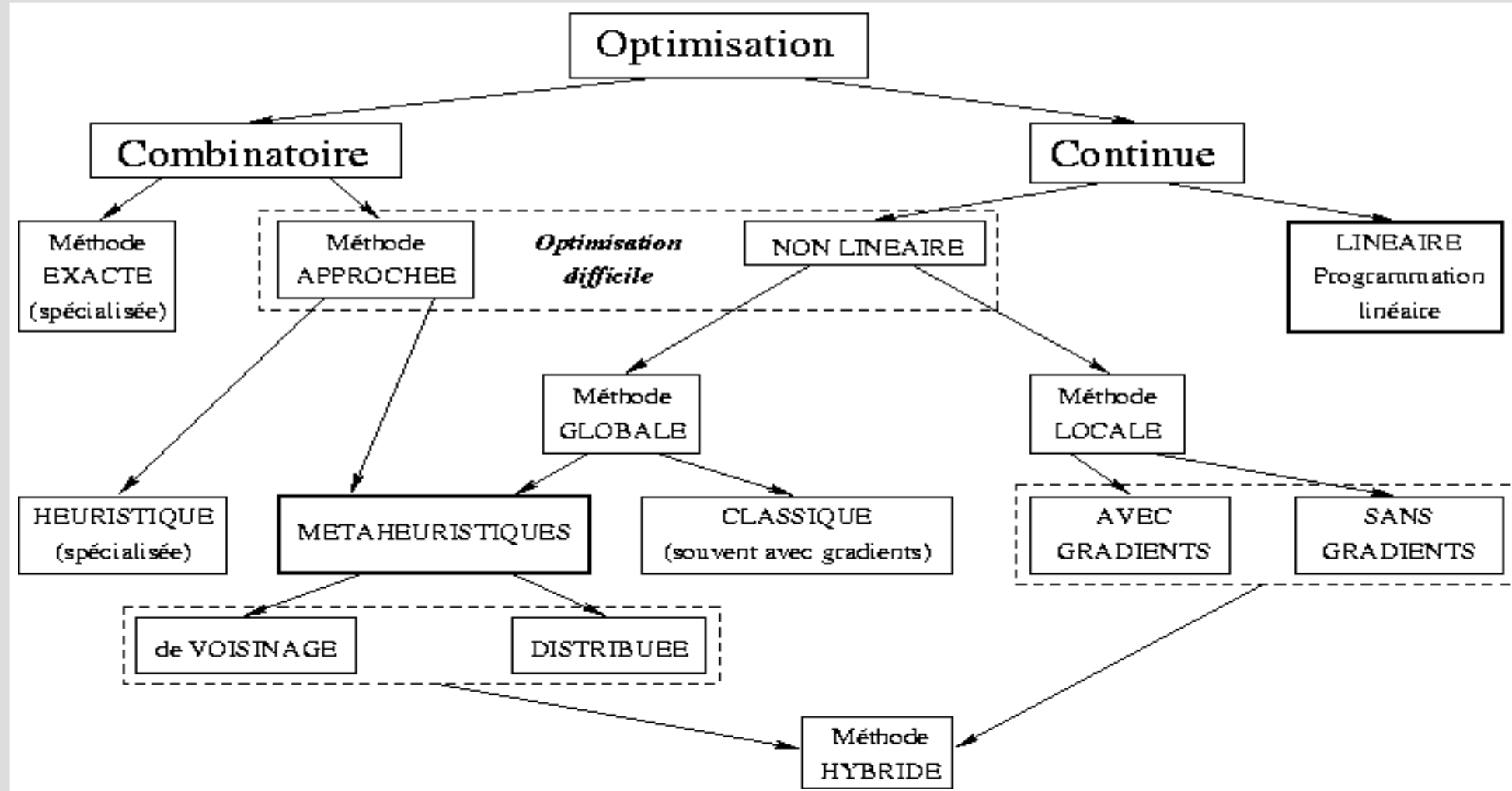
Troisième partie :

Résolution (approchée) de  
Programmes Linéaires  
en Nombres Entiers

# Résolution « approchée » ?

- Résoudre un P(L)NE peut prendre longtemps !
- Mais si on a besoin d'une solution rapidement ?
  - Idée : rechercher une « bonne » solution
    - Solution admissible (borne primale)
    - Pas nécessairement optimale, mais meilleure que la plupart des autres solutions admissibles !
    - Intérêt : recherche rapide (si possible, cf PVC) !
- Ces sol. (et les algorithmes pour les rechercher) sont dit(e)s approché(e)s, ou **heuristiques**
- Possibilité de validation par des bornes duales

# Panorama des méthodes de résolution



# Algorithmes approchés

- Différentes méthodes à utiliser :
  - Méthodes par S&E avortées
  - Algorithmes approchés particuliers (adaptés au pb)
    - Méthode gloutonne (Couverture par les sommets)
    - Arrondi de la solution optimale de la RC (SAD)
    - Autre algorithme spécifique
  - Méthodes génériques (métaheuristiques)
    - Recherche tabou
    - Recuit simulé
    - Algorithmes génétiques
    - Autres : colonies de fourmis, etc.

# Méthodes par S&E avortées

- Consistent simplement à exécuter partiellement une méthode par Séparation & Evaluation, et à l'arrêter quand un critère d'arrêt est vérifié :
  - Temps limite (risque = pas de borne primale !)
  - Ecart prédéfini entre borne primale / borne duale
  - Critère mixte : temps limite si borne primale existe OU écart prédéfini entre borne primale/duale
- Avantage : réutilise les méthodes par S&E !
- Inconvénient : réutilise les méthodes par S&E !

# Algorithme approché spécifique : le sac-à-dos (1/3)

- Idée : mettre à 1 toutes les variables qui valent 1 dans la solution optimale de la RC (et à 0 les autres) fournit-il une bonne solution entière ?

- Vérifions sur un exemple :

$$\max x + (b-1)y$$

$$x + by \leq b$$

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

- On prend  $x=1$  et  $y=0$ , car  $1/1 > (b-1)/b$  :
  - Valeur de la solution (entière) obtenue = 1
  - Valeur de la solution (entière) optimale =  $b-1$  !

# Algorithme approché spécifique : le sac-à-dos (2/3)

- 2e idée : prendre la meilleure de 2 sol. simples
  - 1ère sol. : la solution précédente (**arrondi** de la RC)
  - 2e sol. : choisir le 1er objet non inclus dans le SAD
  - Solution admissible, facile à calculer
- Analyse du pb  $\{\max \sum_i c_i x_i : \sum_i a_i x_i \leq b, x_i \in \{0, 1\} \forall i\}$  :
  - Soit  $j$  la dernière variable à 1 dans la 1ère solution :  
1ère sol. de valeur  $\sum_{i=1 \text{ à } j} c_i$  et 2ème sol. de valeur  $c_{j+1}$
  - Solution approchée de valeur  $\max\{\sum_{i=1 \text{ à } j} c_i, c_{j+1}\}$

# Algorithme approché spécifique : le sac-à-dos (3/3)

- La RC fournit une borne duale, donc :
  - Val. opt. SAD  $\leq \sum_{i=1 \text{ à } j} c_i + x_{j+1}^* c_{j+1} \leq \sum_{i=1 \text{ à } j+1} c_i$
  - Or,  $\sum_{i=1 \text{ à } j} c_i + c_{j+1} \leq 2 \max(\sum_{i=1 \text{ à } j} c_i, c_{j+1})$
- Avec cette variante, la valeur de la solution admissible calculée est au moins la moitié de la valeur d'une solution optimale !
- « Garantie de performance » a priori :  
l'algorithme peut être bien meilleur en pratique !

# Algorithme approché spécifique : couverture par les sommets

- Approche **gloutonne** :
  - Tant que le graphe  $G=(S,A)$  possède des arêtes faire :
    - Choisir une arête au hasard et inclure ses deux extrémités  $u$  et  $v$  dans la couverture  $C$  en construction
    - Supprimer cette arête et les arêtes adjacentes à  $u$  ou  $v$
- On a au plus  $|A|$  itérations, et :
  - $C$  est bien une couverture, car une arête n'est supprimée de  $G$  que si elle est « couverte »
  - Les  $|C|/2$  arêtes choisies sont disjointes par les sommets, donc la taille d'une solution optimale est au moins  $|C|/2 =$  la moitié de la taille de la sol.  $C$  calculée

# A retenir...

- Avantages des algo. approchés spécifiques:
  - Exploitent les particularités du problème
  - Parfois, analyse théorique de l'algo. possible
  - Peuvent être utilisés **avant** une métaheuristique
- Inconvénients des algo. approchés spécifiques :
  - Exploitent les particularités du pb = non portables !
  - Contrairement aux méthodes par S&E avortées, la conception d'un tel algorithme reste délicate quand trouver une borne primale est difficile (PVC)

# Métaheuristiques

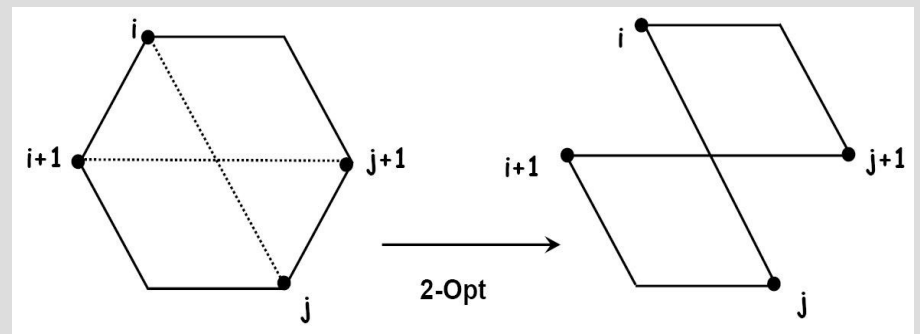
- Heuristique = algorithme approché
- Métaheuristique = algo. approché « générique »
- Basées sur plusieurs principes communs :
  - Voisinage, recherche locale, etc.
- Schéma commun :
  - Un algorithme générique (souvent) stochastique
  - Un ensemble de paramètres, qui permet d'adapter l'algorithme à chaque problème particulier

# Fondements

- Rechercher rapidement une ou des solutions « localement optimales »
- Notion d'optimum local :
  - Une solution est un optimum local, ou localement optimale, si elle est meilleure que toutes les autres solutions d'un voisinage prédéfini
  - Voisinage ?
    - Voisins d'une solution = solutions « proches » d'elle
    - Proximité ?
      - Notions exactes à définir selon les besoins...

# Voisinage et transformations

- En fait, le voisinage d'une solution donnée  $x$  est l'ensemble des solutions admissibles obtenues à partir de  $x$  via une transformation élémentaire
  - Définir le voisinage = définir la/les transformation(s) !
  - Voisinage défini a priori : la transformation dépend du pb considéré (adaptation du schéma générique)
  - La transformation doit être suffisamment simple
    - Sac-à-dos : échanger (au plus)  $k$  objets
    - PVC : voisinage  $k$ -opt



# Optimalité locale vs globale (1/2)

- Par définition, un optimum global est un optimum local (quel que soit le voisinage)
- Par contre, un optimum local n'est pas nécessairement un optimum global !

– Exemple de sac-à-dos ( $x_i \in \{0, 1\}$  pour tout  $i$ ) :

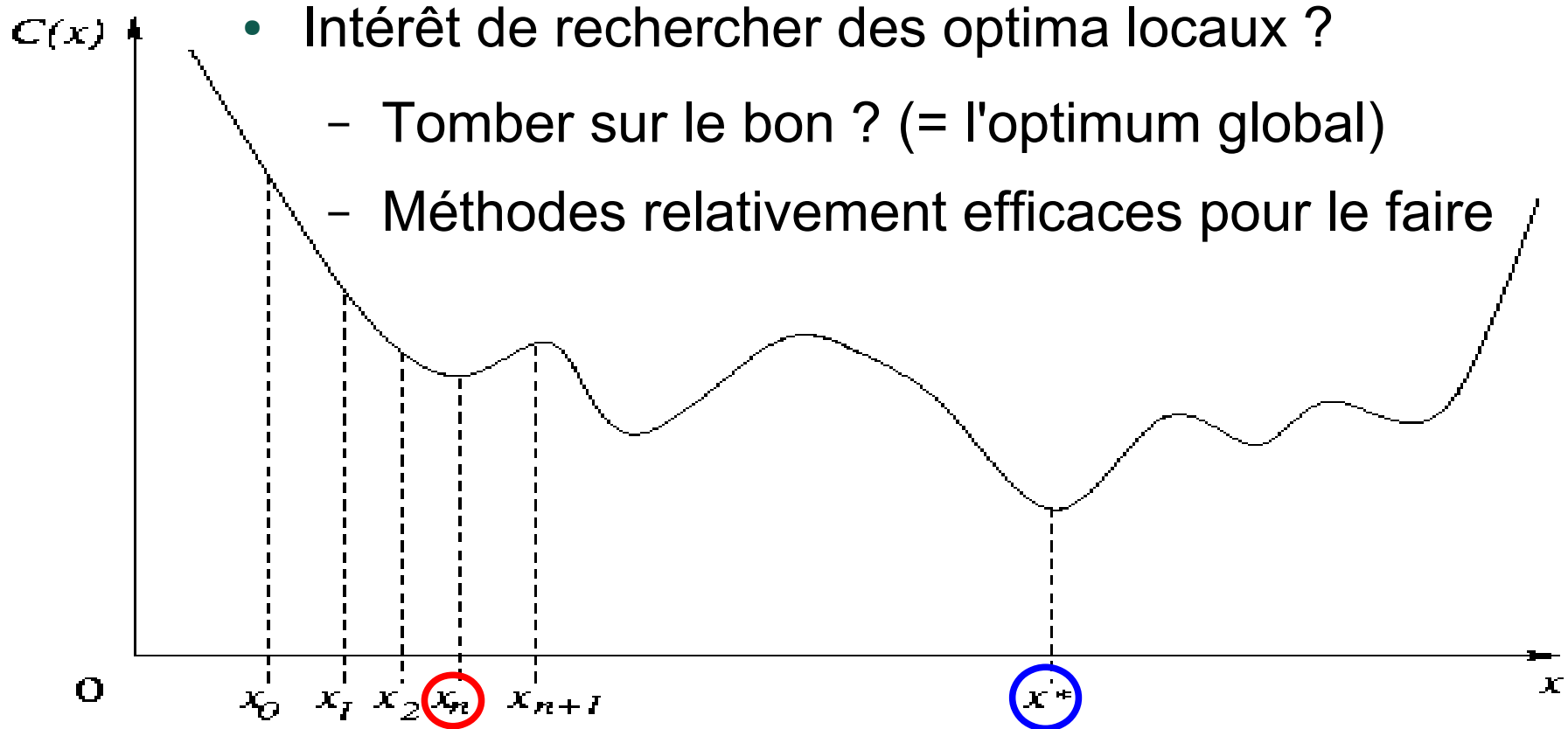
$$\max 15x_1 + 18x_2 + 4x_3 + 2x_4 + 5x_5 + x_6$$

$$3x_1 + 4x_2 + x_3 + x_4 + 3x_5 + x_6 \leq 5$$

– Transformation (voisinage) : échanger deux objets

$$(x_1 = x_3 = x_4 = 1, x_2 = x_5 = x_6 = 0) = \text{optimum local, pas global !}$$

# Optimalité locale vs globale (2/2)



Piège à éviter : rester bloqué dans un optimum **local**, loin du **global**

Pour cela, il faut bien définir le voisinage ET réussir à s'extraire du voisinage d'un optimum local

# Trouver un optimum local : méthode par recherche locale (RL)

- $f$ =fonction objectif (on minimise)
- $V(x)$ =voisinage (voisins admissibles) de la sol.  $x$
- Algorithme de recherche locale par descente :
  - Choisir une solution initiale  $x$
  - Tant que  $x$  n'est pas un minimum local faire
    - Choisir un  $x' \in V(x)$  tel que  $f(x') < f(x)$
    - $x := x'$
  - Fin tant que
- En pratique, on peut décider de s'arrêter quand on a trouvé  $x' \in V(x)$  tel que  $f(x') < f(x)$

# S'extraire d'un optimum local ?

- S'autoriser à dégrader la valeur de la fonction objectif (méthodes par RL/de voisinage)
  - Avec une certaine probabilité, à contrôler
    - Recuit simulé
  - Si, après « apprentissage », aucun autre mouvement n'est possible
    - Méthode/recherche tabou
- Faire évoluer plusieurs solutions en parallèle (et donc explorer simultanément plusieurs zones)
  - Algorithmes distribués/évolutifs (génétiques, etc.)

# Quelle métaheuristique choisir ?

- Quelle est la plus efficace ?
  - Aucune, et toutes à la fois : cela dépend du pb !
  - En « moyenne », toutes ont la même efficacité : seul résultat théorique réellement applicable
- Comment en choisir une, pour un pb donné ?
  - Pas de règle absolue pour :
    - Choisir une métaheuristique
    - Régler ses paramètres
  - Expérience (savoir-faire) et/ou expérimentations (comparaison expérimentale) !

# Mise en oeuvre de métaheuristiques

- Réglage des paramètres = adaptation au pb :
  - Définir une solution initiale
  - Définir une structure de voisinage
  - Définir les paramètres spécifiques
- Bon usage : comparer bornes primales et duales !
- Critère d'arrêt :
  - Temps limite
  - Ecart de la valeur de la sol. courante à la borne duale
  - Combinaison de ces deux critères

# Recherche tabou (RT) - *Tabu search* (1/4)

- Méthode proposée par F. Glover en 1986
- Principe : associer un mécanisme de mémoire à une recherche locale classique
- Plus précisément, on construit une liste T (***liste tabou***) contenant les  $p$  dernières sol. visitées
  - $p$  dépend du pb/des instances considéré(es)
- A partir de la solution courante, on parcourt le voisinage, et on se déplace vers le meilleur voisin, tout en modifiant la liste tabou au besoin

# Recherche tabou (RT) - *Tabu search* (2/4)

- Description formelle de l'algorithme (on min.) :
  - Trouver une solution initiale  $x$  ;  $x^* := x$  ;  $T := \emptyset$  ;
  - Tant que critère d'arrêt non vérifié faire
    - $x := \operatorname{argmin}_{x' \in V(x)} f(x')$  (RL sur le voisinage non tabou de  $x$ )
    - $T := T \cup \{x\}$  (éliminer si besoin le plus ancien élément de  $T$ )
    - Si  $f(x) < f(x^*)$  alors
      - $x^* := x$  ( $x^*$  est la meilleure solution courante)
    - Fin si
  - Fin tant que

# Recherche tabou (RT) - *Tabu search* (3/4)

- Paramètre (choix concepteur) : taille  $p$  de la liste tabou  $T$ =file FIFO (typiquement,  $p \in [5, 10]$ )
- En pratique, on stocke les transformations dans  $T$ , et non les solutions complètes
  - Efficacité (temps, espace) vs perte d'information
- Capacité de la RT à s'extraire des min. locaux ?
  - Si  $\min_{x' \in V(x)} f(x') > f(x)$ , on « remonte » (on effectue une transformation qui accroît la fonction objectif  $f$ )
  - Ce mécanisme est susceptible de permettre d'échapper à un minimum local

# Recherche tabou (RT) - *Tabu search* (4/4)

- Algorithme déterministe ou stochastique ?
  - Dépend du choix de la solution initiale (aléatoire ?)
  - Variante stochastique : on cherche un meilleur voisin, et non le meilleur
- Conditions d'arrêt :
  - Conditions déjà évoquées
  - $V(x)$  vide (rare)
  - Nb maximum d'itérations sans améliorations

# Recuit simulé - RS (1/4)

- Méthode de *Simulated Annealing* proposée par Kirkpatrick, Gelatt et Vecchi en 1983
- Principe : échapper aux minimums locaux en autorisant des transformations dégradant la fonction objectif avec une certaine probabilité
  - Loi de Gibbs-Boltzmann car analogie avec le phénomène thermodynamique de recuit des métaux
- Cette probabilité dépend :
  - Du taux de dégradation de la fonction objectif
  - D'un param. diminuant avec le temps (*température*)

# Recuit simulé - RS (2/4)

- Description formelle de l'algorithme (on min.) :
  - $x := \text{sol. initiale}$  ;  $x^* := x$  ;  $T := \text{température initiale}$  ;
  - Tant que condition d'arrêt non vérifiée faire
    - Tant que non fin de palier faire
      - Choisir  $x'$  dans  $V(x)$  ;  $\Delta f := f(x') - f(x)$  ;
      - Si  $(\Delta f < 0)$  alors  $x := x'$  ;
      - Sinon tirer  $p$  dans  $[0, 1]$  ; Si  $(p \leq \exp(-\Delta f/T))$  alors  $x := x'$  Fin si
      - Fin si
      - Si  $f(x) < f(x^*)$  alors  $x^* := x$  Fin si
    - Fin tant que
    - $T := g(T)$  ;
  - Fin tant que

# Recuit simulé - RS (3/4)

- Réglages des paramètres ?
  - Solution initiale : cf recherche tabou (aléatoire ?)
  - Initialisation de la température : règle de Kirkpatrick...
  - $g(T)=kT$ , avec  $0 < k < 1$  (en pratique,  $k \in [0.85, 0.95]$ )
  - Fin de palier (palier trop long=convergence lente, palier trop court=minimum local) ?
    - Nb d'itérations prédéfini/fixe
    - Nb minimum de transformations « coûteuses »
  - Condition d'arrêt :
    - Nb maximum de paliers fixé à l'avance
    - Écart par rapport à une borne duale

# Recuit simulé - RS (4/4)

- Bilan de la méthode :
  - Pas de mémoire (contrairement à la RT)
  - Structurellement et « massivement » stochastique
  - Beaucoup, beaucoup de paramètres à régler !
  - Très simple à adapter même pour des pb ayant des structures très différentes
  - Souvent plus lente que la RT, mais parfois avec des gains en performance (qualité des solutions obtenues) significatifs

# A retenir... (1/2)

- Avantages des métaheuristiques :
  - Souvent efficaces en pratique
  - Pas d'algo. spécifique à concevoir pour chaque pb
  - N'utilisent pas la formulation P(L)NE, donc indépendantes d'une éventuelle non linéarité
  - La diversité des métaheuristiques existantes offre une certaine souplesse, car si une métaheuristique est inefficace sur un problème donné, on peut en essayer d'autres, avec une mise en oeuvre rapide
  - Peuvent être utilisées **avant** une méthode par S&E

# A retenir... (2/2)

- Inconvénients des métaheuristiques :
  - Garantie de performance a posteriori (la plupart du temps, pas d'analyse théorique possible)
  - Nécessité de régler des paramètres spécifiques aux pb par des expérimentations (pas d'anticipation)
  - Comme pour les algorithmes approchés spécifiques, difficiles (impossibles ?) à utiliser si trouver une borne primale (solution initiale) est difficile pour le pb

# Programmation Linéaire en Nombres Entiers

Quatrième partie :

Cas « faciles » en  
Programmation Linéaire  
en Nombres Entiers

# Existence de cas « faciles » ?

- Déjà vu : si la sol. optimale de la RC d'un PLNE est entière, c'est une solution entière optimale
  - C'est un cas « facile », car on peut résoudre le PLNE à l'aide de l'algorithme du simplexe (vs S&E)
- Question (problème !) : trouver des conditions pour garantir une telle situation ? Difficultés :
  - Cela peut dépendre de la fonction objectif ! (Ex. 1)
  - Une solution optimale entière peut exister, mais ne pas être trouvée par le simplexe ! (Ex. 2)

# Reconnaissance des cas « faciles » ?

- Exemple 1 :  $P = \{(x_1, x_2) : x_1 \in [1, 7/3], x_2 \in [2, 5/2]\}$ 
  - (a)  $\max x_1 + x_2$  s.c.  $(x_1, x_2) \in P$  ?  $(7/3, 5/2)$  : non entière !
  - (b)  $\min x_1 + x_2$  s.c.  $(x_1, x_2) \in P$  ?  $(1, 2)$  : entière !
- Exemple 2 :  
 $\max\{x_1 + x_2 \text{ s.c. } x_1 \geq 0.5, x_2 \geq 0.5, x_1 + x_2 \leq 2\}$ 
  - Solutions de base optimales :  $(0.5, 1.5), (1.5, 0.5)$
  - Solution optimale entière :  $(1, 1)$ , de valeur 2 aussi...
    - Pas une solution de base, donc pas trouvée par le simplexe !

# Condition suffisante ? (1/2)

- Comment garantir un cas :
  - « Facile » et
  - Qu'on puisse reconnaître comme tel ?
- En fait, si TOUS les sommets du polyèdre sont entiers, alors la solution optimale trouvée par le simplexe le sera nécessairement !
- **Un tel polyèdre est dit polyèdre à sommets entiers, ou simplement polyèdre entier**
  - Donc, polyèdre entier = problème « facile » (CS)

# Condition suffisante ? (2/2)

- Mais un polyèdre (et donc son éventuel caractère entier) est lié au PLNE modélisant le problème !
  - Ex. : les polyèdres  $P = \{x_1 + x_2 \leq 1, x_1 \geq 0, x_2 \geq 0\}$  et  $Q = \{2x_1 + 4x_2 \leq 5, x_1 \in [0, 1], x_2 \geq 0\}$  contiennent les mêmes solutions entières
  - Le problème est facile si on le résout sur  $P$ , mais pas sur  $Q$  !
- Et comment prouver qu'un polyèdre est entier ?
  - Enumérer tous ses sommets, et les tester ?
  - Complicé en général, mais envisageable si
    - Peu de variables/contraintes OU
    - Méthode générale disponible : CS ?

# Totale unimodularité

- Notion fournissant une CS = totale unimodularité (d'une matrice associée à un polyèdre donné)
  - Une matrice  $M$  à coefficients  $0, 1, -1$  est dite **totale unimodulaire** (TU) ssi toute sous-matrice carrée de  $M$  a comme déterminant  $1, -1$  ou  $0$
- Théorème : si la matrice associée à un polyèdre  $P$  est TU et si le second membre des contraintes est entier, alors toute solution de base de  $P$  est entière (autrement dit,  $P$  est entier)
  - Pas CN, car par ex.  $\{2x_1 + 3x_2 \leq 6, x_1 \geq 0, x_2 \geq 0\}$  est entier

# Application des matrices TU : sélection d'intervalles (1/3)

- Un problème de sélection de tâches :
  - Un ensemble de  $n$  tâches : chaque tâche est représentée par un intervalle de temps  $[d_i, f_i]$
  - On dispose d'un seul employé pour les exécuter
  - 2 tâches qui se chevauchent sont **incompatibles**
  - On veut sélectionner (pour les exécuter) un nombre maximum de tâches compatibles
    - Modélisation du problème ?
      - Problème du stable (sous-ensemble de sommets non reliés entre eux) maximum dans un graphe d'intervalles (donc triangulé)
    - Résolution du problème ?

# Application des matrices TU : sélection d'intervalles (2/3)

- Modélisation PLNE ?
  - Une variable 0-1 par tâche
  - Une contrainte par paire de tâches incompatibles ?
  - Alternative : découpage du temps en créneaux
    - On note  $e_1, \dots, e_k$  les  $d_i, f_i$  classés par ordre croissant ( $k \leq 2n$ )
    - Une contrainte (au plus une tâche exécutée) par créneau  $[e_i, e_{i+1}] \Rightarrow$  matrice avec au plus  $2n-1$  lignes
    - Une tâche traverse des créneaux consécutifs !
  - La matrice des contraintes obtenue est TU, donc le PLNE se résout comme un PL !

# Application des matrices TU : sélection d'intervalles (3/3)

- Méthode de résolution alternative ?
  - En fait, comme beaucoup de PLNE avec matrice TU, il y a une méthode de résolution simple pour ce pb
    - Une matrice TU est souvent un « indice » que le pb considéré est « facile »
  - Algorithme glouton spécifique :
    - On suppose les tâches triées par  $f_i$  croissants
    - Tant qu'il reste des tâches à sélectionner faire
      - Sélectionner la tâche de plus petit indice qui soit compatible avec les tâches déjà sélectionnées
    - Fin tant que

# A retenir...

- Les PLNE avec matrices TU constituent une famille de polyèdres entiers, et donc une famille de problèmes « faciles »
  - Lien intéressant entre PL et PLNE
  - Lien avec des pb importants (voir la suite)
- Souvent, il existe aussi des algo. combinatoires efficaces pour les pb ayant des matrices TU
  - Les résultats sur les matrices TU ne donnent malheureusement pas d'indications sur les méthodes à utiliser si le polyèdre n'est pas entier...

# Programmation Linéaire en Nombres Entiers

Cinquième partie (facultative) :

Méthodes de coupes pour la  
Programmation Linéaire  
en Nombres Entiers

# Enveloppe convexe des solutions d'un PLNE (1/2)

- Soit un PLNE : l'enveloppe convexe de ses sol. (entières) admissibles est un polyèdre (entier !)
- Trouver la « meilleure » solution entière du PLNE = trouver le « meilleur » sommet du polyèdre
- Il y a donc équivalence entre résoudre le PLNE et résoudre le PL dont le polyèdre est donné par l'enveloppe convexe des solutions entières
- Si on peut « décrire » cette enveloppe convexe, on peut donc résoudre le PLNE via la PL !

# Enveloppe convexe des solutions d'un PLNE (2/2)

- Comment faire ? Est-ce difficile ?
  - Malheureusement, pas de méthode miracle...
- Problèmes posés :
  - Lien entre formulation PLNE et enveloppe convexe des solutions entières pas nécessairement évidente !
    - Difficulté pour décrire l'enveloppe convexe
  - Le nb de contraintes nécessaires pour la description de cette enveloppe peut être (trop) important
    - Difficulté pour résoudre le (gros) PL obtenu

# Inégalités valides

- On appelle **inégalité valide** une contrainte linéaire vérifiée par toute sol. entière d'un PLNE
  - Une telle contrainte permet (idéalement) de « couper » des sol. continues, c'est-à-dire d'effectuer une **coupe** dans le domaine admissible de la RC
    - Inégalités ajoutées dans un algorithme par S&E non valides
- Idée : ajouter suffisamment d'inégalités valides à la RC pour obtenir l'env. conv. des sol. entières
  - Toute inégalité valide permet-elle de décrire une partie de cette enveloppe convexe ?

# Inégalités valides fortes

- Certaines inégalités valides ne font pas partie de la « frontière » de cette enveloppe convexe !
- Trouver des inégalités valides ne suffit pas : il faut trouver les « bonnes » !
- Inégalité valide **forte** = qui appartient à la frontière de l'enveloppe convexe
- Une inégalité valide forte définit une **facette** (un des hyperplans composant la frontière) du polyèdre des solutions entières

# Couplage maximum (1/2)

- PLNE du problème du couplage maximum :
  - Dans un graphe  $G$ , trouver un ensemble d'arêtes disjointes par les sommets qui soit de taille maximum
    - Variables 0-1 (une variable par arête)
    - Au plus une arête du couplage adjacente à  $s$ ,  $\forall s \in S$
- Description de l'env. conv. des sol. entières ?
  - Au plus une arête du couplage adjacente à  $s$ ,  $\forall s \in S$
  - Variables positives
  - (1) Pour tout  $S' \subseteq S$  avec  $S' \neq \emptyset$  et  $|S'|$  impair, il y a au plus  $(|S'|-1)/2$  arêtes ( $i \in S', j \in S'$ ) dans le couplage

# Couplage maximum (2/2)

- Th. : les inégalités (1) sont valides
- Th. : les contraintes listées suffisent pour décrire l'enveloppe convexe des solutions entières
- Nombre de contraintes (1) à considérer ?
  - Considérer tous les sous-ensembles de sommets  $S'$  de cardinalité impaire, soit de l'ordre de  $\approx 2^{|S|-1}$
- Ici, il existe des méthodes pour résoudre le PL efficacement **sans expliciter l'ensemble des contraintes (1)** : ce n'est pas toujours le cas !

# Coupes/troncatures de (Chvatal-)Gomory (1/2)

- Méthode proposée en 1958 par Ralph Gomory
- Idée : étant donnée une solution optimale non entière de la RC, ajouter des inégalités valides pour rendre cette solution non admissible
- Principe de mise en œuvre : utiliser le tableau du simplexe associé à la solution à « couper » pour générer une inégalité valide qu'elle ne vérifie pas
  - En pratique, on choisit une variable  $x_i$  non entière dans la solution, et on génère une inégalité pour  $x_i$

# Coupes/troncatures de (Chvatal-)Gomory (2/2)

- Description formelle et analyse :
  - Ligne  $i$  du tableau optimal :  $x_{B(i)} + \sum_{j \in HB} t_{ij} x_j = \underline{b}_i$  ( $\underline{b}_i \notin \mathbf{Z}$ )
  - $x_i \geq 0 \ \forall i \Rightarrow x_{B(i)} + \sum_{j \in HB} \lfloor t_{ij} \rfloor x_j \leq \underline{b}_i$  (où  $\lfloor 1.7 \rfloor = \lfloor 1 \rfloor = \lfloor -0.1 \rfloor = -1$ )
  - $x_i$  entière  $\forall i \Rightarrow x_{B(i)} + \sum_{j \in HB} \lfloor t_{ij} \rfloor x_j \leq \lfloor \underline{b}_i \rfloor$
  - On a donc  $\sum_{j \in HB} (t_{ij} - \lfloor t_{ij} \rfloor) x_j \geq \underline{b}_i - \lfloor \underline{b}_i \rfloor$ 
    - C'est bien une inégalité valide !
    - La solution continue courante ne la vérifie pas (car  $x_j = 0$  pour tout  $j$  dans  $HB$ , mais  $\underline{b}_i - \lfloor \underline{b}_i \rfloor > 0$  puisque  $\underline{b}_i$  non entier)

# Méthode alternative de résolution (exacte) d'un PLNE

- Possibilité d'utiliser ces coupes dans l'algorithme de résolution de PLNE suivant (alternative à S&E)
  - Résoudre la RC du PLNE
  - Tant que solution optimale non entière faire
    - Calculer une coupe à partir du tableau optimal
    - Résoudre la RC du PLNE
  - Fin tant que
- Convergence garantie si coupes bien choisies
- Si PLVM au lieu de PLNE : plus compliqué...

# Exemple d'algorithme utilisant les coupes polyédrales

- On veut résoudre le PLNE suivant :  
$$\max\{21x_1 + 11x_2 \text{ s.c. } 7x_1 + 4x_2 + x_3 = 13, x_i \in \mathbf{IN} \text{ pour tout } i\}$$
  - Sol. opt. de la RC :  $x_1 = 13/7$  et  $x_2 = x_3 = 0$ , de valeur 39
- Au bout de 2 itérations (ajout d'une contrainte, puis résolution de la RC), on obtient :
  - Solution optimale de la 3e RC :  $x_1 = 0$ ,  $x_2 = 3$  et  $x_3 = 1$
  - Solution entière de valeur 33, et donc optimale pour le PLNE initial

# A retenir...

- Couper, une alternative à la résolution par S&E
  - Coupes de (Chvatal-)Gomory pour le simplexe
- En réalité, on peut combiner méthodes par S&E et ajout d'inégalités valides :
  - Méthode de coupes & branchements (Branch & Cut)
  - En chaque nœud de l'arbre de recherche, on résout la RC à laquelle on a ajouté un certain nombre d'inégalités valides
  - C'est la méthode de résolution utilisée par CPLEX !

# LA P(L)NE : BILAN

- Permet de modéliser de très nombreux pb
- Résolution d'un PLNE plus difficile qu'un PL, et nécessite de faire un compromis temps/qualité
  - Méthodes exactes (méthodes par S&E, coupes) :
    - Résolution à l'optimum
    - Parfois trop lentes
  - Méthodes approchées ((méta-)heuristiques) :
    - Qualité des solutions variable (et souvent non garantie)
    - Peuvent être très rapides si bien utilisées
  - Quelques cas particuliers « faciles » (lien avec la PL)

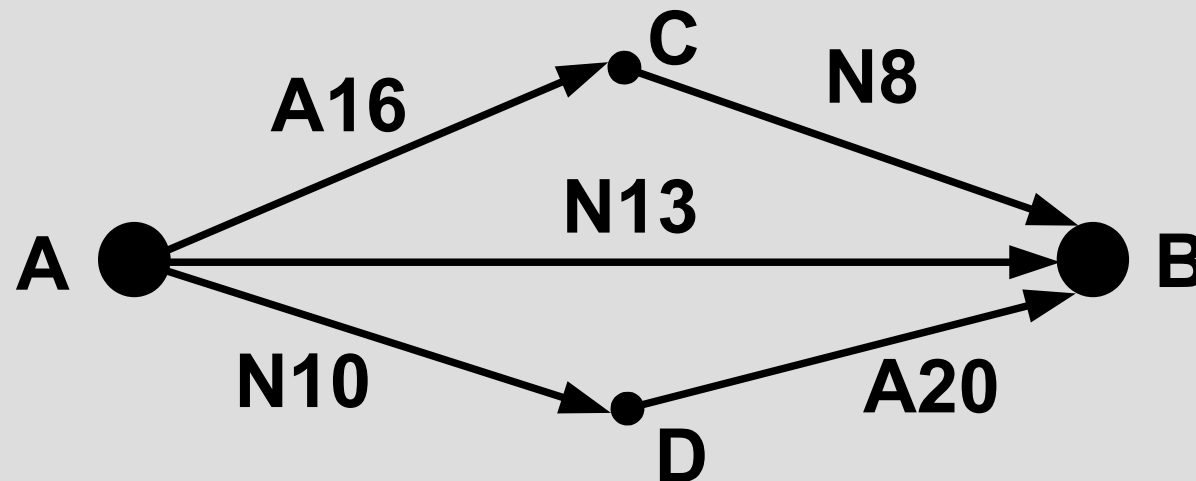
# Problèmes de chemins et de flots

Première partie :

Chemins optimaux & application  
à l'ordonnancement de projets

# Un premier exemple (1/2)

- Un individu désire se rendre d'une ville A à une ville B le plus rapidement possible
  - 3 trajets possibles :
    - Prendre l'Autoroute A16, puis la Nationale 8 (N8) à C
    - Prendre la Nationale 13 (N13)
    - Prendre la Nationale 10 (N10), puis l'Autoroute A20 à D



# Un premier exemple (2/2)

- Son GPS indique les temps de parcours suivants :
  - A16 : 10 minutes le matin, 15 minutes le soir
  - N8 : 7 minutes le matin, 5 minutes le soir
  - N13 : 18 minutes le matin, 19 minutes le soir
  - N10 : 12 minutes le matin, 18 minutes le soir
  - A20 : 5 minutes le matin, 3 minutes le soir
- Ces temps varient en fonction de l'heure, et ne dépendent donc pas uniquement des distances
  - Trajet du matin : A16+N8 ou N10+A20 (17 minutes)
  - Trajet du soir : N13 (19 minutes)

# Problèmes de plus courts chemins (PCC)

- Plus généralement, étant donnés :
  - Un **graphe** représentant un **réseau** (routier/ferroviaire, électrique, télécom, de relations entre éléments, etc.)
  - Des **longueurs** (représentant des distances, des temps, etc.) sur les liens (arcs) de ce graphe  $G$
  - Un sommet **origine**  $o$ , un sommet **destination**  $d$  de  $G$
- Comment déterminer un plus court chemin  $o \rightarrow d$  ?
  - Plus court chemin de  $o$  à  $d$  = chemin (trajet) de  $o$  à  $d$  de longueur minimum parmi tous les chemins de  $o$  à  $d$
  - Longueur d'un chemin = somme des long. de ses arcs

# Problèmes de chemins optimaux

- En fait, plusieurs types de problèmes ont un sens :
  - Plus **court** chemin
    - Applications (longueurs  $\geq 0$ ) : GPS, tables de routage, etc.
  - Plus **long** chemin
    - Problème du plus court chemin avec longueurs  $\geq 0$  = problème du plus long chemin avec longueurs  $\leq 0$
- En général, pas d'algorithme efficace connu
  - Quelques cas particuliers pertinents ET « faciles »
    - Plus court chemin avec longueurs  $\geq 0$
    - Chemin optimal (plus court/long) dans un graphe sans circuit

# Plus courts chemins (PCC) avec longueurs positives

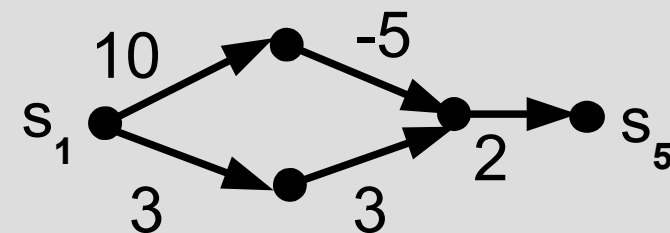
- Pour ce cas, on utilise l'algorithme de **DIJKSTRA**
  - On associe une valeur  $d_i$  à tout sommet  $s_i$  ( $0=s_1$ ,  $d=?$ ), on note  $\text{long}_{ij}$  la longueur de l'arc  $(s_i, s_j)$ , et on a alors :
    - Initialement, aucun sommet n'est traité,  $d_1=0$  et  $d_i=+\infty$  si  $i > 1$   
(A la fin,  $d_i$ =longueur d'un plus court chemin de  $s_1$  à  $s_i$ )
    - Pour  $p$  allant de 1 au nombre de sommets faire
      - Choisir le sommet  $s_i$  non traité avec  $d_i=\min_j\{d_j$  pour les  $s_j$  non traités}
      - Traiter le sommet  $s_i$  : pour tout  $j$  tel qu'il existe un arc  $(s_i, s_j)$ , poser  $d_j=\min\{d_j, d_i+\text{long}_{ij}\}$

# Analyse de l'algorithme de DIJKSTRA (1/2)

- On note  $\text{dist}(s_1, s_i)$  la longueur d'un PCC de  $s_1$  à  $s_i$ 
  - A tout instant, on a, pour tout  $i$ ,  $\text{dist}(s_1, s_i) \leq d_i$
  - Si on trie les  $s_i$  par ordre de  $\text{dist}(s_1, s_i)$  croissants, alors, à l'itération  $i$ ,  $d_j = \text{dist}(s_1, s_j)$  si  $j \leq i$  et on traite le sommet  $s_i$ 
    - Vrai pour  $i=1$  (trivial)
    - Montrons que c'est vrai pour  $i+1$  si c'est vrai pour  $i$ 
      - Le prédécesseur de  $s_{i+1}$  dans tout PCC entre  $s_1$  et  $s_{i+1}$  est un des  $i$  sommets  $s_1, \dots, s_i$ , qui sont, par hypothèse, les  $i$  premiers sommets traités (vrai uniquement car toutes les longueurs sont positives !!!)
      - On a donc bien  $d_{i+1} = \text{dist}(s_1, s_{i+1}) \leq \text{dist}(s_1, s_j) \leq d_j$  pour tout  $j > i+1$ , et  $s_{i+1}$  est par conséquent le sommet traité à l'itération  $i+1$

# Analyse de l'algorithme de DIJKSTRA (2/2)

- En réalité, cet algorithme fournit donc la longueur d'un plus court chemin de  $s_1$  à  $s_i$ , pour tout  $i$ 
  - Pour tout arc  $(s_i, s_j)$ , si on a  $\text{dist}(s_1, s_j) = \text{dist}(s_1, s_i) + \text{long}_{ij}$ , alors  $(s_i, s_j)$  fait partie de l'**arborescence des plus courts chemins** (choix arbitraire en cas d'égalité)
    - Arbre de racine  $s_1$ , dans lequel, pour tout  $i$ , (unique) chemin de  $s_1$  à  $s_i =$  plus court chemin de  $s_1$  à  $s_i$  dans le graphe initial
- Analyse fausse s'il existe des longueurs  $\leq 0$ 
  - Exemple où l'algorithme échoue ?
    - $\text{dist}(s_1, s_5) = 7$ , mais l'algo. trouve 8



# Plus courts chemins avec longueurs négatives

- En réalité, il existe d'autres algorithmes efficaces (cf Floyd) même en présence :

- De longueurs  $\leq 0$  (+ courts chemins)
- De longueurs  $\geq 0$  (+ longs chemins)

```
Algorithme FLOYD  
dij = longij pour tous i,j  
Pour tous k,i,j faire  
dij = min(dij, dik + dkj)  
(version PCC)
```

- Néanmoins, même de tels algorithmes échouent en présence de **circuits absorbants**
  - Circuit de longueur négative (plus courts chemins)
  - Circuit de longueur positive (plus longs chemins)
    - Diminuer/augmenter indéfiniment la longueur de tout chemin
  - Une CS pour éviter ce cas : graphes sans circuits...

# Chemins optimaux dans les graphes sans circuits (GSC)

- Trouver dans un graphe  $G$  sans circuits dont les arcs ont des longueurs quelconques un plus court/ plus long chemin de l'origine  $o$  à la destination  $d$ 
  - Idée : traiter les  $n$  sommets dans un certain ordre...
  - Concept : **numérotation** (ou **ordre**) **topologique**
    - Un numéro  $i$  entre 1 et  $n$  associé à chaque sommet  $s_i$
    - Pour tout arc  $(s_i, s_j)$ , on a  $i < j$
    - Un tel ordre existe ssi le graphe est sans circuits :
      - Si un tel ordre existe, clairement aucun circuit ne peut exister
      - Si aucun circuit n'existe, alors il y a un sommet sans prédécesseur. On peut le numéroté 1, puis le supprimer, et recommencer...

# Algorithme pour les chemins optimaux dans les GSC

- Description de l'algorithme (plus longs chemins)
  - Déterminer un ordre topologique sur les  $n$  sommets, numérotés de  $s_1$  à  $s_n$  (avec  $s_1=0$ )
  - Initialement,  $d_1=0$  et  $d_i=-\infty$  si  $i > 1$  (à la fin,  $d_i$ =longueur d'un plus long chemin de  $s_1$  à  $s_i$ )
  - Pour  $i$  allant de 1 à  $n$  faire
    - Traiter le sommet  $s_i$  : poser
$$d_i = \max_j \{d_j + \text{long}_{ji}, \text{ pour } j \text{ tel qu'il existe un arc } (s_j, s_i)\}$$
(Et si on cherche des plus courts chemins ?)

# Analyse de l'algorithme dans les GSC (Bellman)

- Similarités avec l'algorithme de Dijkstra :
  - Traiter les  $s_i$  dans un certain ordre, une fois chacun
  - Arborescence des plus courts/plus longs chemins issus de  $o$  définie de façon similaire
- Optimalité de l'algorithme ?
  - A l'étape  $i$ ,  $d_i$  = longueur d'un plus long chemin de  $s_1$  à  $s_i$ 
    - Vrai pour  $i=1$  (trivial)
    - Montrer que c'est vrai pour  $i+1$  si c'est vrai pour  $i$  ?
      - Tout prédécesseur de  $s_{i+1}$  dans un plus long chemin issu de  $o$  est un des  $i$  sommets  $s_1, \dots, s_i$ , donc on a  $d_{i+1} = \max_{j \text{ prédécesseur de } i+1} \{d_j + \text{long}_{ji+1}\}$

# Application des chemins optimaux à l'ordonnancement de projets (1/2)

- Soit le problème d'**ordonnancement** suivant :
  - Un projet constitué d'un ensemble de  $n$  **tâches**
    - Chaque tâche  $t_i$  (ou opération) a une durée d'exécution  $p_i$
  - Pour chaque tâche  $t_i$ , on a un ensemble de tâches (qui peut être vide) qui doivent avoir été exécutées avant  $t_i$ 
    - Contraintes de précédence
      - Tâches préalables, précédence partielle, contraintes de date
  - Planifier les tâches en minimisant la durée du projet ?
    - Contraintes non prises en compte
      - Ressources : supposées suffisantes (ou extensibles)
      - Tâches incompatibles (non exécutables simultanément)

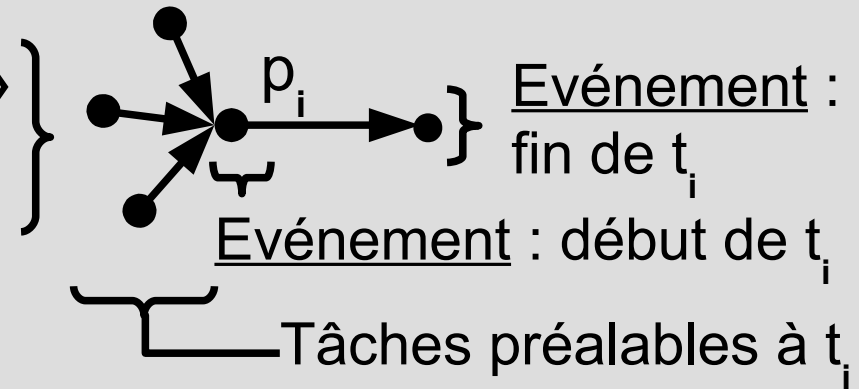
# Application des chemins optimaux à l'ordonnancement de projets (2/2)

- On peut associer deux modèles à ce problème
  - Modèle (US) PERT (*Project Evaluation and Review Technique*), variante de la CPM (*Critical Path Method*)
  - Modèle MPM (*Méthode des Potentiels Métra*)
    - Méthode française développée par B. Roy, et publiée en 1958 dans sa revue Métra
- Deux modèles, mais des fondements communs
  - Modéliser les tâches et leurs contraintes via un graphe
    - La construction du graphe diffère selon la méthode choisie !
  - Déterminer la durée minimale d'un ordonnancement = calculer **un plus long chemin** dans ce graphe

# Description du modèle PERT

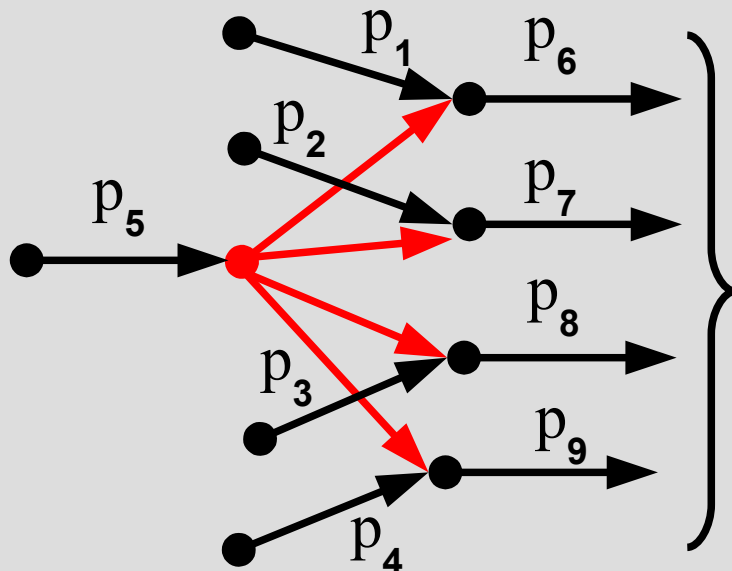
- Modèle (ou graphe) dit « événements-tâches »

- Sommets = « événements »
- Arcs = tâches



- Inconvénient :

- Ajout éventuel de nombreuses tâches « fictives » !

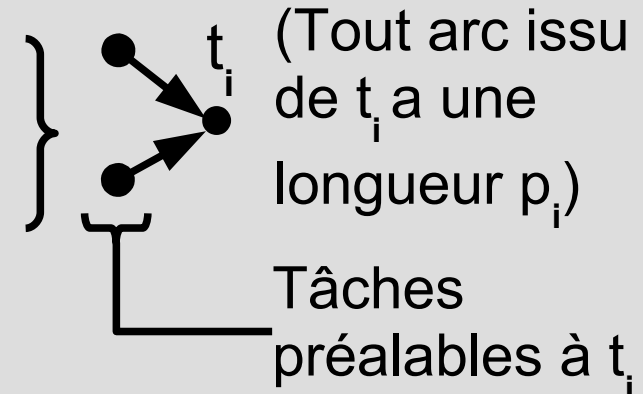


Pour tout  $i < 5$ , la tâche  $t_{i+5}$  a comme tâches préalables les tâches  $t_i$  et  $t_5$  : pour garantir ces contraintes de précédence, il est donc nécessaire d'ajouter 4 tâches **fictives** de durée d'exécution nulle

# Description du modèle MPM

- Modèle (ou graphe) dit « tâches-précédences »

- Sommets = tâches
- Arcs = contraintes de précédence



- Avantages :

- Construction du graphe MPM (à partir des contraintes de précédence) en général simple et évidente
- Pas de tâches fictives à introduire, sauf 2 (de durée 0)
  - Une tâche « Début », préalable à toute tâche initialement sans tâche préalable (sommet sans prédécesseur)
  - Une tâche « Fin », ayant pour tâche préalable toute tâche qui initialement n'est préalable à aucune autre tâche

# Résolution à l'aide de MPM (1/3)

- Comment déterminer une date de début et une date de fin pour chaque tâche, de façon à :
  - respecter les (3 types de) contraintes de précédence,
  - minimiser la durée totale du projet (date de fin) ?
- Ceci est possible ssi le MPM obtenu est un GSC
  - CN : s'il existe un circuit, on est en présence de contraintes de précédence incompatibles...
  - CS : étant donné un tel GSC, on va relier la durée de l'ordonnancement (**durée du projet**) associé à un **calcul de plus long chemin dans ce GSC**

# Résolution à l'aide de MPM (2/3)

- 4 valeurs associées à chaque tâche (noeud)  $t_i$  :
  - Date de début **au plus tôt** (à partir d'où  $t_i$  peut débuter)
    - $T_i = \max_j \{T_j + p_j, \text{ pour } t_j \text{ tâche préalable à } t_i\}$
  - Date de début **au plus tard** (jusqu'où  $t_i$  peut débuter)
    - $T_i^* = \min_j \{T_j^* - p_i, \text{ pour } t_j \text{ ayant } t_i \text{ pour tâche préalable}\}$
  - Marge **totale** (retard sans impact sur la durée du projet)
    - $MT_i = T_i^* - T_i$
  - Marge **libre** (retard sans impact sur les dates au + tôt)
    - $ML_i = \min_j \{T_j - T_i - p_i, \text{ pour } t_j \text{ ayant } t_i \text{ pour tâche préalable}\}$

# Résolution à l'aide de MPM (3/3)

- **Tâche critique**  $t_i$  :
  - Retard sur  $t_i$  = retard identique sur la durée du projet
  - On a  $T_i^* = T_i$ , soit  $MT_i = 0$  (par exemple, la tâche « Fin »)
- Un **chemin critique** du graphe MPM est un chemin d'origine « Début » et de destination « Fin », composé uniquement de tâches critiques : c'est un plus long chemin dans ce GSC
  - Calcul des dates au plus tôt : cf algorithme de Bellman
  - Durée minimum du projet = longueur de ce chemin

# A retenir...

- Problèmes d'OC intéressants à plusieurs titres
  - Admettent des modèles simples (graphes)
  - Admettent des algorithmes de résolution efficaces
    - Dijkstra (plus courts chemins avec longueurs positives)
    - Bellman (chemins optimaux dans les GSC)
  - Admettent diverses applications immédiates
    - Routage, GPS
    - Planification de projets (ordonnancement)
- Possibilité de généralisation
  - Problèmes multi-chemins...

# Problèmes de chemins et de flots

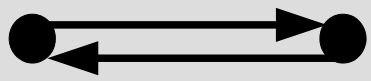
Deuxième partie :

Flots maximums  
et coupes minimums

# Gérer un réseau de distribution d'eau : énoncé du problème

- On désire approvisionner un client en eau
  - Ce client est relié par branchement à un réseau de distribution d'eau, incluant un château (d'eau)
  - Ce réseau est constitué de canalisations (tuyaux) reliées entre elles par des noeuds de raccordement
  - Les canalisations ont différents diamètres : le diamètre d'une canalisation détermine son débit maximum
  - Peut-on acheminer l'eau dans ce réseau de façon à ce que le débit d'eau arrivant au client soit au moins égal à un certain débit minimum contractuel ?
    - Exemple ?

# Gérer un réseau de distribution d'eau : modèle et variante

- Modèle (on peut interdire  ) :
  - Un graphe représentant le réseau : un sommet par noeud, un arc muni d'une **capacité** par canalisation
  - Origine (source) = château, destination (puits) = client
    - Réseau de transport : graphe avec source, puits, capacités
    - Débit circulant dans un tuyau donné = **flot** sur l'arc associé
    - Toute l'eau qui sort du château d'eau parvient au client
- On peut aussi être amené à traiter le problème d'approvisionnement de plusieurs clients...

# Formalisation du problème

- Problème du flot (débit) maximum :
  - On cherche en fait à maximiser le débit d'eau émis par la source (château d'eau)
    - Tout le volume d'eau émis par la source parvient au client
  - Toute l'eau parvenant à un noeud en repart
    - **Conservation du flot (à chaque noeud)**
  - On doit tenir compte des débits maximums (diamètres)
    - **Contraintes de capacité (sur les canalisations)**
- Modèle PLNE ?
  - Une variable  $f_{ij} \geq 0$  par arc  $(i,j)$  : flot sur l'arc  $(i,j)$

# Propriétés de la formulation

- Matrice du PLNE du flot maximum :
  - Dans les contraintes de conservation du flot, chaque variable  $f_{ij}$  (flot sur l'arc  $(i,j)$ ) apparaît au plus 2 fois :
    - Avec coefficient +1, dans la contrainte (ligne) associée à  $j$
    - Avec coefficient -1, dans la contrainte (ligne) associée à  $i$
    - Sauf pour les arcs incidents à la source ou au puits
  - Chaque autre contrainte n'implique qu'une variable
    - Contrainte inutile si on modifie le graphe convenablement ?
- La matrice est **TU** : le problème est donc « facile »
  - Autre algorithme efficace ?

# Notion de graphe d'écart

- Algorithme de Ford & Fulkerson pour ce PL(NE)
  - Algorithme de « marquage » efficace (1956), qui peut aussi se décrire à l'aide de la notion de **graphe d'écart**
- Etant donné un réseau de transport  $R$  et un flot  $f = (f_{ij})_{ij \text{ est un arc}}$ , le graphe d'écart associé à  $R$  et  $f$  est
  - Défini sur le même ensemble de sommets  $S$  que  $R$
  - Pour tout arc  $(i,j)$  de capacité  $c_{ij}$  dans  $R$ , muni :
    - d'un arc  $(i,j)$  de capacité  $c_{ij}$  si  $f_{ij} = 0$
    - d'un arc  $(j,i)$  de capacité  $c_{ij}$  si  $f_{ij} = c_{ij}$  (l'arc  $(i,j)$  est dit **saturé**)
    - d'arcs  $(i,j)$  et  $(j,i)$  de capacités  $c_{ij} - f_{ij}$  et  $f_{ij}$  (respectivement) sinon

# Algorithme de Ford & Fulkerson

- Description de l'algorithme (pour un réseau  $R$ ) :
    - Calculer un **flot initial** pour  $R$ , et en faire le flot courant (la méthode gloutonne fournit un flot **complet**, où il y a un arc saturé sur tout chemin de la source au puits)
    - Déterminer le graphe d'écart associé au flot initial
    - Tant qu'il existe un chemin de la source au puits dans le graphe d'écart associé au flot courant
      - Choisir un tel chemin, puis calculer la capacité minimum  $\delta$  de ses arcs, et, pour tout arc  $(i,j)$  de ce chemin dans le graphe d'écart, poser  $f_{ij} := f_{ij} + \delta$  si l'arc  $(i,j)$  est dans  $R$  et  $f_{ij} := f_{ij} - \delta$  sinon
      - Déterminer le graphe d'écart associé au flot courant
- (On peut montrer que l'algorithme termine.)

# Notion de coupe dans un graphe

- Coupe dans un graphe avec une source  $s$  et un puits  $p$  = ensemble d'arcs dont la suppression ne laisse aucun chemin entre  $s$  et  $p$ 
  - Valeur de la coupe = somme des capacités de ses arcs
  - Support de la coupe = sommets restant liés à  $s$
- Propriété : valeur de tout flot de  $s$  à  $p$  = (valeur du flot « sortant » de  $U$ ) - (valeur du flot « entrant » en  $U$ ), pour tout  $U \subset S$  connexe contenant  $s$  et pas  $p$ 
  - Conséquence (+ contraintes de capacité) : la valeur de tout flot est inférieure à la valeur de toute coupe

# Analyse de l'algorithme

- Preuve de l'optimalité de cet algorithme :
  - On considère le dernier graphe d'écart
    - Pas de chemin entre  $s$  et  $p$  dans ce graphe
    - Soit  $U$  l'ensemble des sommets reliés à  $s$  dans ce graphe
      - Tout arc  $(i,j)$  de ce graphe d'écart ayant une et une seule extrémité dans  $U$  est tel que  $j$  est dans  $U$  et  $i$  n'y est pas
      - Ainsi, de par la définition des graphes d'écart, soit  $(i,j)$  est un arc du réseau initial  $R$  et  $f_{ij}=0$ , soit  $(j,i)$  est un arc de  $R$  et  $f_{ij}=c_{ij}$
    - Autrement dit,  $U$  est le support de la coupe constituée des arcs de la forme  $(i,j)$  avec  $i$  dans  $U$  et  $j$  pas dans  $U$ 
      - La somme des flots circulant dans ces arcs  $(i,j)$  est donc à la fois la valeur du flot entre  $s$  et  $t$ , et la valeur de la coupe associée
      - **On a un flot et une coupe de même valeur, et donc optimaux**

# Théorème de Ford & Fulkerson et lien avec la dualité en PL

- Théorème flot maximum-coupe minimum (F&F) : valeur d'un flot max. = valeur d'une coupe min.
- Preuve alternative : les PL(NE) de ces deux problèmes sont duaux (au sens de la PL)

$$\left\{ \begin{array}{l} \min \sum_{(i,j) \text{ est un arc}} c_{ij} y_{ij} \\ y_{ij} \geq y_i - y_j, \forall (i,j) \\ y_s = 1, y_p = 0 \\ y_{ij} \geq 0, \forall (i,j) \end{array} \right. \iff \begin{array}{l} y_i = 1 \text{ ssi le sommet } i \text{ est relié à } s : \\ \text{ainsi, l'arc } (i,j) \text{ est coupé ssi } y_{ij} = 1 \end{array}$$

# Flot à coût minimum (1/3)

- Variante : un **coût unitaire** de circulation par arc
  - On veut alors approvisionner le puits à moindre coût
  - Modèle (graphe) : un réseau (graphe), un coût unitaire et une capacité par arc, une source  $s$ , un puits  $p$
- Formalisation : problème du flot à coût minimum
  - Minimiser la somme totale des coûts des arcs
    - Coût d'un arc = quantité de flot sur l'arc fois son coût unitaire
  - Contrainte de capacité
  - Conservation du flot
  - Quantité de flot connue  $F$  à acheminer de  $s$  à  $p$

# Flot à coût minimum (2/3)

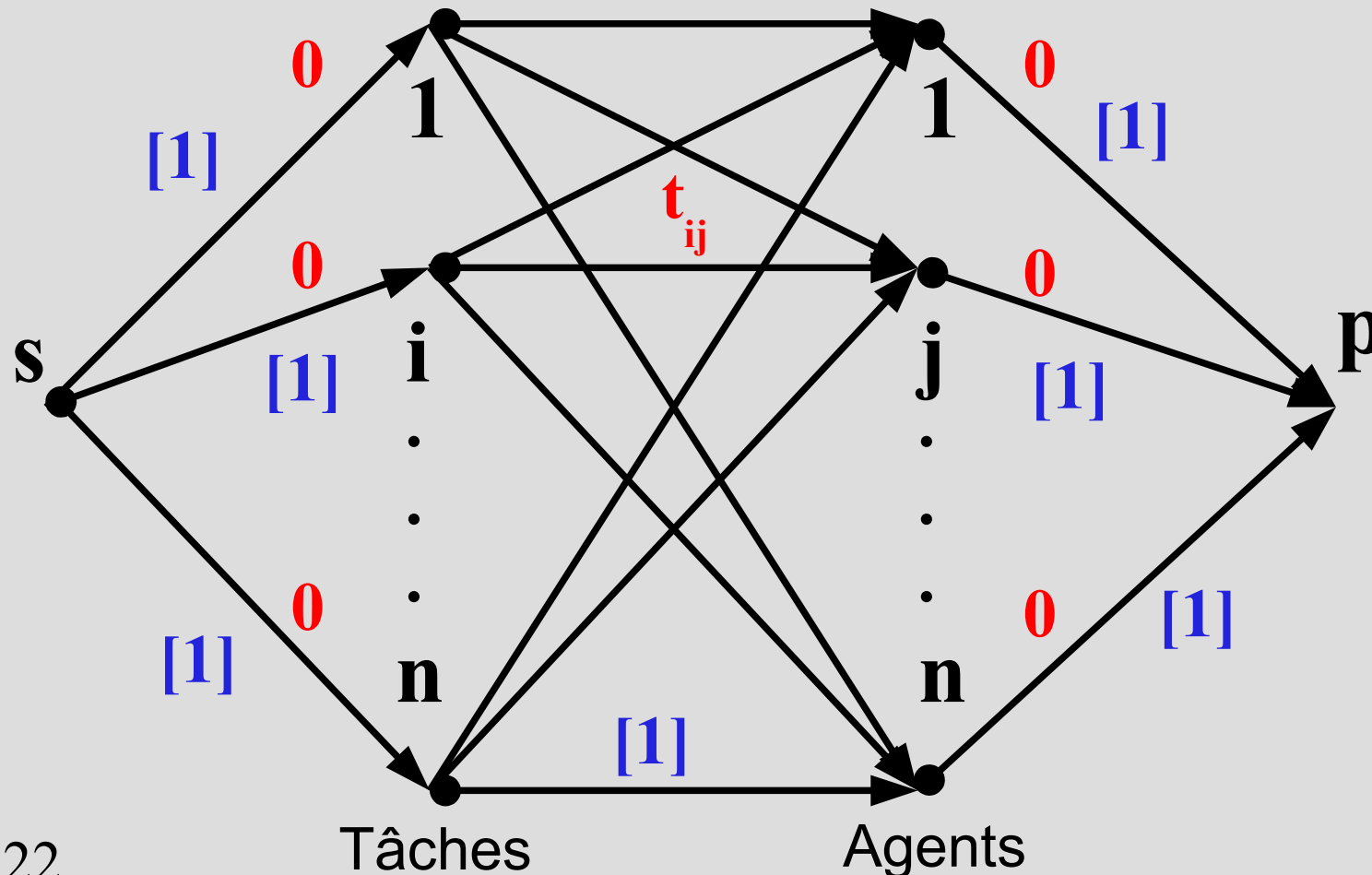
- Beaucoup de similitudes avec d'autres problèmes
  - Si  $F =$  valeur d'un flot maximum (cf algorithme de Ford & Fulkerson) : flot maximum à coût minimum
    - En particulier, la matrice du problème est également TU
  - Si  $F = 1$  ? Problème de plus court chemin de  $s$  à  $p$  !
- Méthode de résolution similaire (graphes d'écart)
  - Algorithme de Busacker & Gowen (cf la suite)
  - La longueur d'un arc  $(i,j)$  dans le graphe d'écart associé à un réseau de transport  $R$  et à un flot  $f$  est :
    - Le coût unitaire de  $(i,j)$  si l'arc  $(i,j)$  est dans  $R$
    - L'opposé du coût unitaire de  $(i,j)$  sinon

# Flot à coût minimum (3/3)

- Propriétés du graphe d'écart d'un flot  $f$  de valeur  $V$ 
  - $f$  est à coût minimum parmi les flots de valeur  $V$  ssi le graphe d'écart associé n'a pas de circuit absorbant
- Algorithme pour le flot (maximum) à coût minimum
  - Légère variante de l'algorithme de F. & F. : quand on choisit un chemin de  $s$  à  $p$ , on prend un + court chemin
    - Dijkstra (longueurs  $\leq 0$ ) et Bellman (circuits) inutilisables...
- Choix du flot initial à coût minimum ?
  - Méthode gloutonne malheureusement inutilisable...
  - Le plus simple : partir du flot nul !

# Lien entre flot à coût minimum et affectation linéaire

- Soit un problème d'affectation linéaire donné par une matrice à  $n$  lignes et  $n$  colonnes  $T=(t_{ij})$



[capacité]  
coût

Flot à coût minimum de valeur  $n$  = affectation optimale !

# A retenir...

- Modèles PL(NE) des problèmes de flot : flot maximum, flot à coût minimum (lien avec les PCC)
- Notion de graphes d'écart et leur utilisation
  - Algorithme de Ford & Fulkerson
  - Algorithme de Busacker & Gowen
- Notion de coupe et théorème de Ford & Fulkerson
- Applications : affectation linéaire, routage de paquets dans un réseau IP, placement optimal (cf TD : placement de tâches, etc.)

