

Recherche Opérationnelle

Sandra U. Ngueveu

Toulouse INP N7 - LAAS/CNRS

<https://homepages.laas.fr/sungueve>

<https://moodle-n7.inp-toulouse.fr/course/view.php?id=2380>

2A SN B-M-L : 2024/2025

Recherche Opérationnelle

Introduction

Sandra U. Ngueveu

Toulouse INP N7 - LAAS/CNRS
<https://homepages.laas.fr/sungueve>

<https://moodle-n7.inp-toulouse.fr/course/view.php?id=2380>

2A SN B-M-L : 2024/2025

Exemple : Planification de maintenance appliquée au réseau ferroviaire français

Optimisation des tournées d'inspection des voies ferrées - SNCF.

Les maintenances préventives sont des **auscultations ultrasoniques** du champignon du rail, qui renseignent sur l'apparition de nouvelles **fissures dans le rail** et permettent de suivre leur évolution. Les fréquences de passages sur les tronçons de voies varient de **6 mois à 10 ans** en fonction principalement du tonnage annuel qui les traverse. Cette maintenance préventive est réalisée à l'aide de trains spécialisés qui surveillent plus de **50.000 km de voies**. Ces engins ne sont pas autorisés à ausculter toutes les parties du réseau, pour des raisons techniques, celles-ci sont auscultées manuellement.

L'usure des voies étant accélérée par la constante augmentation du volume de trafic, les fréquences de surveillance doivent augmenter pour assurer la fiabilité du réseau d'où un **impact important sur le coût**.

Exemple : Planification de maintenance appliquée au réseau ferroviaire français

Optimisation des tournées d'inspection des voies ferrées - SNCF.

- Variables de décision
 - Affectation des tâches aux engins
 - Dates de réalisation
- Fonction-Objectif : minimiser
 - la somme des durées des transferts
- Contraintes : données par le cahier des charges
 - Fréquences d'auscultation
 - Caractéristiques des engins
 - ...

Exemple : Planification de maintenance appliquée au réseau ferroviaire français

Cahier des Charges : différents types de contraintes

- Caractéristiques des engins :
 - vitesse en auscultation, vitesse en transfert
 - autonomie en eau nécessaire à l'auscultation
 - rdvs obligatoires de maintenance
 - tronçons compatibles
- Contraintes de ressource
 - interdiction d'auscultation simultanée par plusieurs engins d'une même région...
- Demandes de ressources des régions qui précisent
 - les gares et les dates au plus tôt de prise en charge / restitution au plus tard de l'engin

Exemple : Planification de maintenance appliquée au réseau ferroviaire français

- Ordre de magnitude du problème

- 40000 tronçons
- 30000 sommets
- Horizon temporel = 365 jours
- Durée des tâches variant de 30 min à 5 jours

Objectifs du cours

Donner un aperçu de la démarche et d'outils de recherche opérationnelle afin de trouver la meilleure solution parmi un ensemble de solutions possibles pour un problème donné.

L'ensemble des solutions possibles n'est pas donné explicitement mais défini par un ensemble de contraintes : cela relève donc de l'optimisation sous contraintes.

Ces contraintes expriment par exemple les contraintes technologiques, les quantités limitées de ressources disponibles (matérielles, temporelles, humaines, financières, . . .)

Recherche opérationnelle - Aide à la décision (RO-AD)

Définition de la RO (source : livre blanc de la R.O.) : Mise en oeuvre de méthodes scientifiques, essentiellement mathématiques, en vue de prendre la meilleure décision possible.

Domaines : Mathématiques + Algorithmique + Informatique (programmation) + Economie

Outils : prog. linéaire, opt. comb., graphes, simulation, files d'attente, théorie des jeux, ...

Les outils de RO-AD :

- aident à trouver :
 - une solution où l'homme n'en trouvait pas
 - plusieurs solutions là où l'homme n'en envisageait qu'une
 - une solution sur des problèmes nouveaux où l'homme n'a aucune expérience
- aident à juger de la qualité d'une solution

Communautés (francophones) des chercheurs et industriels en R.O.

Société Française de R.O. et A.D. (ROADEF) : www.roadef.org

- FORUM : stages, thèses, offres d'emploi

GdR RO (CNRS) : <http://gdrro.lip6.fr/?q=node/229>

Livre blanc de la R.O. : http://www.roadef.org/pdf/LIVRE_BLANC_A5_juin.pdf

Recherche opérationnelle et "Data analytics"

Des données

- Origines :
 - Capteurs, Téléphones portables, Sites Web,
- Types :
 - Séries numériques, images, textes, vidéos, ...
- Accessibilité :
 - De plus en plus de données ouvertes
 - collectivités locales, gouvernement, ...
- Caractéristiques :
 - Explosion de la quantité de données numériques produites
 - Hétérogénéité des données
 - Protection de la vie privée

Recherche opérationnelle et "Data analytics"

3 niveaux de "data analytics"

- Analyse descriptive (Descriptive Analytics)
 - Extraire des connaissances à partir des données
 - Pourquoi y-a-t-il un bouchon ?
- Analyse prédictive (Predictive Analytics)
 - Construire des modèles (pour prévoir le futur)
 - Quel sera le trafic dans 1 heure ?
- Analyse prescriptive (Prescriptive Analytics)
 - Assister la prise de décision
 - Quel est le meilleur itinéraire si je pars à 8 :30 ?

⇒ optimisation / décision / planification

Modélisation

Avant de résoudre un problème, il faut le formaliser, le décrire ; i.e., le modéliser sous une forme connue. Pour cela, il faut identifier :

- les paramètres que l'**on ne peut pas maîtriser** mais dont on connaît la valeur ⇒ les **données**
- les paramètres sur lesquels on peut **agir** ou dont on souhaite **déterminer la valeur** ⇒ les **variables (de décision)**
- ce qui permet de **comparer** les solutions entre elles ⇒ la **fonction objectif** ou le critère
- ce qui **limite** le choix des valeurs de variables de décision ⇒ les **contraintes**

Nous nous focaliserons sur les modèles de type programmes linéaires (PL) et programmes linéaires en nombres entiers (PLNE) :

- Les variables ne peuvent prendre que des valeurs réelles ou entières
- Les contraintes sont linéaires
- La fonction objectif est linéaire

Exemple

Une usine ALPHA produit deux ciments rapportant 50€ et 70€ la tonne. Pour fabriquer une tonne de ciment 1, il faut 40 min de calcination dans un four et 20 min de broyage. Pour fabriquer une tonne de ciment 2, il faut 12 min de four et 30 min de broyage. Le four et l'atelier de broyage sont disponibles 6h et 8h par jour. Quelle quantité de ciment de chaque type peut-on produire par jour pour maximiser le bénéfice ?

Variables

Fonction-objectif

Contraintes

Domaine de définition

Exemple

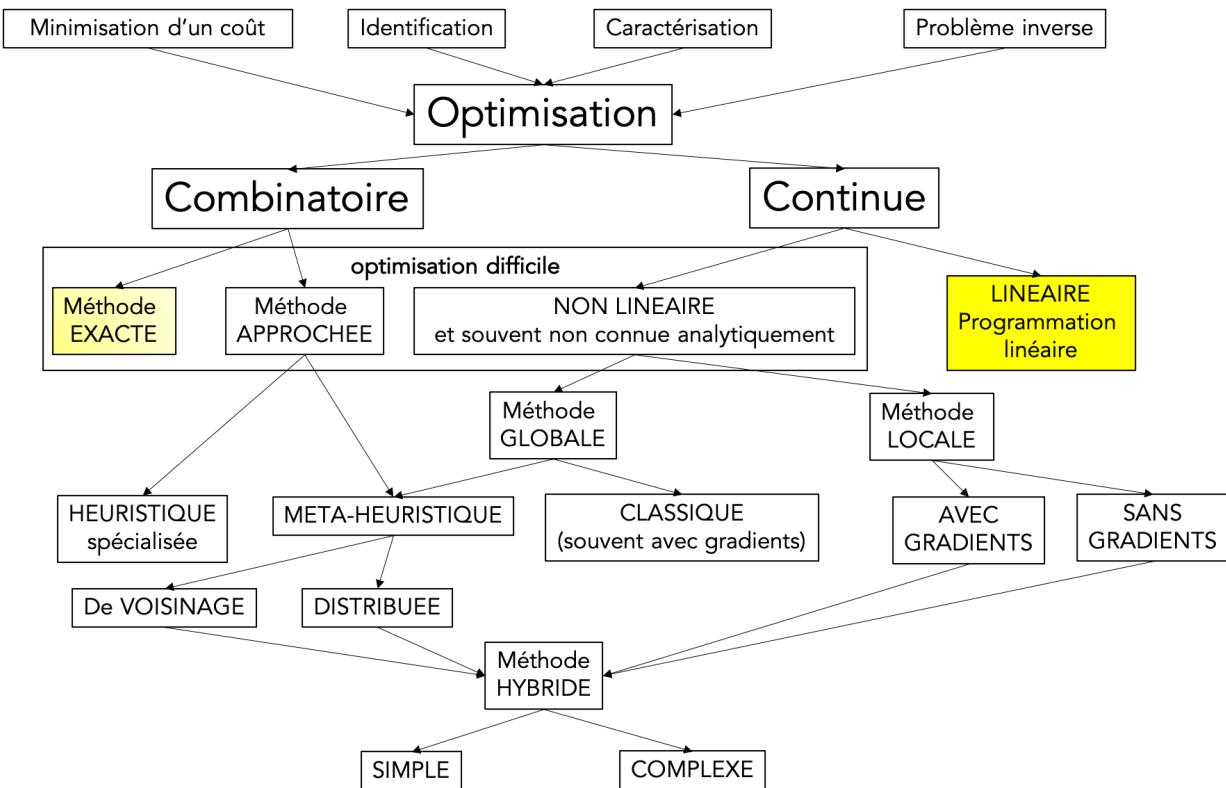
Modèle mathématique résultant

Démarche de modélisation et résolution

Principales étapes de modélisation et résolution d'un problème concret :

- ① **identification du problème** : formulation des objectifs, expressions des choix possibles, recensement des contraintes. La tâche la plus difficile pour cette étape est le recueil des données précises et fiables.
- ② **modélisation** : établir un lien entre les différents paramètres. Utiliser en priorité des modèles connus : programmes linéaires, ou programmes linéaires en nombres entiers (ou mixtes), ou graphes, ...
- ③ **résolution** : utiliser en priorité des méthodes connues pour trouver l'optimum mais également la sensibilité de la solution obtenue par rapport aux imprécisions des données.
- ④ **vérification du modèle** : examiner la solution obtenue. Si elle paraît surprenante (par rapport à l'historique par exemple), vérifier si tous les éléments ont été correctement pris en compte dans le modèle.
- ⑤ **réalisation**. Donner des instructions claires pour l'application de la solution

Classification des méthodes d'optimisation mono-objectif



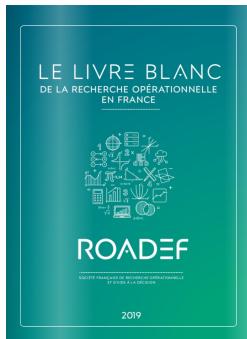
Source : Collette - Siarry 2002

Déroulement et contenu de ce Cours

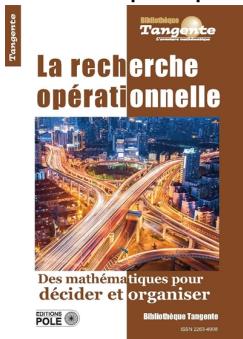
- Bases de programmation linéaire
 - Modélisation
 - Résolution graphique (2D), simplexe primal
- Quelques problèmes classiques en optimisation combinatoire
- Base de programmation linéaire en nombres entiers
 - Modélisation
 - Procédures de séparation et évaluation (Branch-and-bound) : relaxations, calculs de bornes, stratégies de branchement

Quelques ressources “grand public”

Livre Blanc de la R.O.
en France



La Recherche opérationnelle :
des mathématiques pour décider



Des exemples de Tutoriels
sur la chaîne ROADEF



Bulletins ROADEF n°42 :
Ethique des algorithmes



n°43 :
Véhicules autonomes dans la ville



n°44 :
Enjeux de l'optimisation quantique



Sandra U. Ngueveu (N7 - LAAS)

R. O. - support de prise de notes - 2A SN

Bibliographie

2A SN B-M-L : 2024/2025 17 / 18

Bibliographie



Christian Prins et Marc Sevaux (Eyrolles 2011)
Programmation linéaire avec Excel : 55 problèmes d'optimisation modélisés pas à pas et résolus avec Excel



F. Bonnans et S. Gaubert (Les éditions de l'école polytechnique, 2015)
Recherche opérationnelle : Aspects mathématiques et applications



M. Minoux (Lavoisier, 2008)
Programmation mathématique : théorie et algorithmes



K. Mellouli, E. K. Abdelkader et P. Borne (Technip, 2004)
Programmation linéaire et applications



I. Charon, A. Germa et O. Hudry (Masson, 1996)
Méthodes d'optimisation combinatoire partie 1 : programmation linéaire



R. Faure, B. Lemaire, C. Picouleau (Dunod, 2014)
Précis de recherche opérationnelle



Chaire de formation et de recherche "retail responsable" 2022-2027
<https://www.laas.fr/projects/ChaireRetailResp/>

Programmation Linéaire

I. Modélisation et Résolution Graphique en 2D

Sandra U. Ngueveu

Toulouse INP N7 - LAAS/CNRS

<https://homepages.laas.fr/sungueve>

<https://moodle-n7.inp-toulouse.fr/course/view.php?id=2380>

2A SN B-M-L : 2024/2025

Programmation Linéaire (PL)

Modèle Mathématique Linéaire

$$\min(\text{ou } \max) \quad f(X) = c_1x_1 + c_2x_2 + \dots + c_{n-1}x_{n-1} + c_nx_n$$

sous les contraintes (s.c.)

$$a_{11}x_1 + a_{12}x_2 + \dots + a_{1,n-1}x_{n-1} + a_{1n}x_n \geq b_1$$

⋮

$$a_{j,1}x_1 + a_{j,2}x_2 + \dots + a_{j,n-1}x_{n-1} + a_{j,n}x_n = b_j$$

⋮

$$a_{m,1}x_1 + a_{m,2}x_2 + \dots + a_{m,n-1}x_{n-1} + a_{m,n}x_n \leq b_m$$

$$x_i \in \mathbb{R}, \quad i \in [1 \dots n]$$

où les c_i , b_j et a_{ji} sont des coefficients constants.

Programmation Linéaire (PL)

Modèle Mathématique Linéaire

- Les variables :
 - sont en nombre fini n (même s'il est très grand, de l'ordre du million)
 - ne peuvent prendre que des valeurs réelles
- Les contraintes sont linéaires, c'est à dire :
 - sont de type égalité (signe $=$) ou inégalité (signe \leq ou \geq)
 - ont un terme de gauche correspondant à une combinaison linéaire des variables x_i
 - ont un terme de droite égal à une valeur réelle
- La fonction objectif est linéaire

Exemples de Programmes Linéaires

$$\max x_1 + 2x_2 + 3x_3$$

s.c.

$$\begin{aligned} 7x_1 + x_3 &\leq 6 \\ x_1 + 2x_2 &\leq 20 \\ 3x_2 + 4x_3 &\leq 30 \\ x_1 \geq 0, x_2 \in \mathbb{R}, x_3 \geq 0 & \end{aligned}$$

$$\max x_1 + 2x_2 + 3x_3$$

s.c.

$$\begin{bmatrix} 7 & 0 & 1 \\ 1 & 2 & 0 \\ 0 & 3 & 4 \\ -1 & 0 & 0 \\ 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \leq \begin{bmatrix} 6 \\ 20 \\ 30 \\ 0 \\ 0 \end{bmatrix}$$

$$\min -y_1 + y_3$$

s.c.

$$\begin{aligned} y_1 + y_2 &\geq 1 \\ y_1 + 2y_3 &\leq 3 \\ y_1 \in \mathbb{R} & \\ y_2 \in \mathbb{R} & \\ y_3 \geq 0 & \end{aligned}$$

$$\min f(x) = \sum_{i=1}^n \sum_{j=1}^n c_{ij} x_{ij}$$

s.c.

$$\begin{aligned} \sum_{i=1}^n x_{ij} &= 1, \quad \forall j \in [1, \dots, n] \setminus \{i\} \\ \sum_{i=1}^n x_{ji} &= 1, \quad \forall j \in [1, \dots, n] \setminus \{i\} \\ 0 \leq x_{ij} \leq 1, & \quad \forall i \in [1, \dots, n], \forall j \in [1, \dots, n] \end{aligned}$$

où n et les c_{ij} sont des constantes prédéfinies.

Mises en forme particulières

Forme matricielle : On peut noter

- $x = (x_1, x_2, \dots, x_n)^T$ le vecteur des variables
- $b = (b_1, b_2, \dots, b_m)^T$ celui des seconds membres des contraintes,
- $c = (c_1, c_2, \dots, c_n)^T$ les coûts ou profits affectés aux variables
- et A la matrice $m \times n$ des a_{ij} .

Dans ce cas, les deux formes suivantes sont les plus courantes :

Forme Canonique

$$\max c.x \quad \text{ou}$$

$$A.x \leq b$$

$$x \geq 0$$

$$\min c.x$$

$$A.x \geq b$$

$$x \geq 0$$

Forme Standard

$$\max c.x$$

$$A.x = b$$

$$x \geq 0$$

⇒ Comment passer d'une forme à l'autre ?

Un peu d'histoire

La Programmation Linéaire : une branche de la Recherche Opérationnelle

La R.O. apparaît en 1940 en Angleterre puis aux Etats-Unis à des fins de recherche militaire : il s'agit d'utiliser au mieux ses moyens militaires, à l'époque insuffisants (avions, forces antiaériennes (D.C.A.), moyens maritimes).

Naissance de la Programmation Linéaire

En 1947, D.B. Dantzig, venant de soutenir sa thèse et conseiller à l'US Air Force propose l'**algorithme du simplexe** pour résoudre les problèmes de planification des transports lors d'opérations militaires. Avant cela, ces problèmes étaient résolus à la main, sans possibilité de refaire les calculs en cas de changement de dernière minute. De plus, le concept de "fonction-objectif globale" n'existe pas ; les décisions étaient prises par des règles de bon sens sur la base de l'expérience des décideurs.

Méthodes de résolution

Le premier algorithme polynomial pour la programmation linéaire est dérivé de la méthode générale de l'ellipsoïde défini par A. Nemirovski (Prix John von Neumann 2003), D. B. Yudin et N. Shor en 1970. L. Khachiyan a ainsi construit un algorithme de l'ellipsoïde adapté à la programmation linéaire en 1979 dont le mérite tient plus à la contribution en théorie de la complexité et à l'ouverture ainsi réalisée vers les méthodes polynomiales plutôt qu'en son efficacité pratique jugée médiocre. Une nouvelle avancée décisive a été réalisée en 1984 par N. Karmarkar [14], chercheur à IBM qui a proposé, pour la première fois, une **méthode des points intérieurs** dont il a démontré la complexité polynomiale dans le pire des cas.

La majorité des solutions logicielles actuelles sont construites autour de l'algorithme du simplexe et d'algorithmes des points intérieurs.

1 Contexte

2 Représentation, Interprétation et Résolution Graphique du cas 2D

$$\begin{aligned}
 & \text{max} 3x_1 + 2x_2 \\
 \text{s.c. } & 2x_1 + x_2 \leq 18 \\
 & 2x_1 + 3x_2 \leq 42 \\
 & 3x_1 + x_2 \leq 24 \\
 & x_1, x_2 \geq 0
 \end{aligned}$$

Exemple : Représentation

Démarche

- Tracer la droite (D1) correspondant à l'égalité dérivée de la 1ère contrainte
 - Que représentent les points sur la droite ?
 - Quels sont les points qui satisfont la contrainte (1) ?
 - Faire de même avec la contraintes (2)
 - Quels sont les points qui satisfont à la fois les contraintes (1) et (2) ?
 - Faire de même avec la contrainte (3)
 - Faire de même avec le domaine de définition des variables
 - Quels sont les points qui satisfont TOUTES les contraintes du PL ?
- Ces points constituent l'**ensemble des solutions réalisables du PL**.

Exemple : Interprétation

Ensemble des solutions réalisables

Polyèdre (convexe)

Points extremum versus points intérieurs

Exemple : Représentation (2)

- Tracer la droite correspond à une valeur de fonction-objectif de 0 ($z = 0$, ou $f(X) = 0$). Conclure.
- Idem pour $z = 1$.
- Idem pour $z = 3$.
- Identifier le gradient de la fonction-objectif. Conclure.
- Jusqu'où pousser la démarche ?

Récapitulatif de l'approche graphique

Approche Graphique 2D

- Tracer les axes d'abscisses et ordonnées en tenant compte du domaine de définition
- Tracer toutes les contraintes pour obtenir le polyèdre
- Tracer le vecteur-gradient correspondant à la fonction-objectif
- Trouver l'optimum s'il existe

Cas à n variables/dimensions

- Même idée à n dimensions (méthodes de gradient)
- Reste "dessinable" en 3D mais plus compliqué pour $n \geq 4$

Pour conclure ...

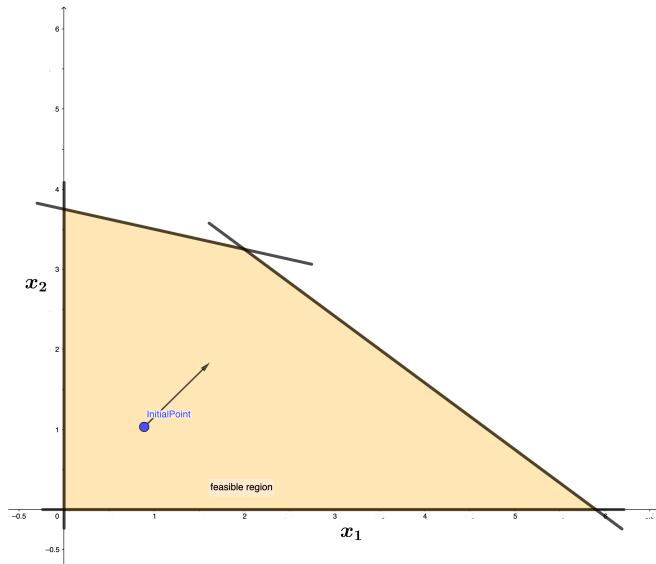
Intérêt/Force de la PL : Optimum local = Optimum global !

Limites : Pour être directement modélisable par PL, les actions/décisions modélisées par les variables doivent être :

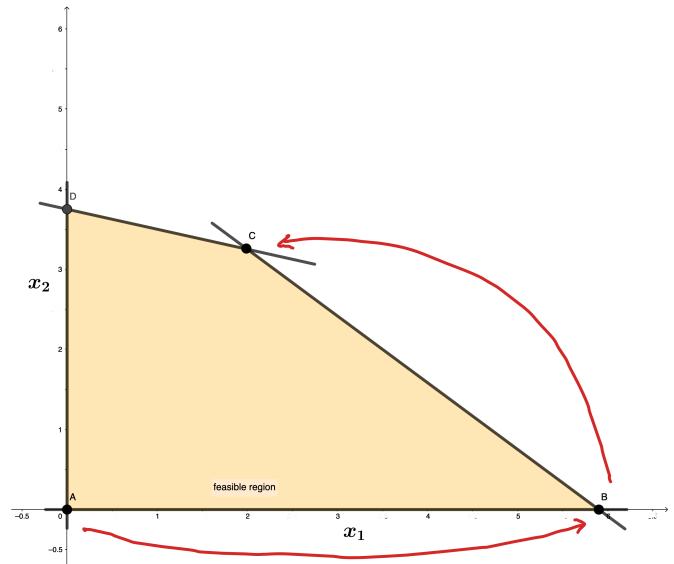
- additives
- proportionnelles
- divisibles

Possibilité d'utiliser des techniques de linéarisation pour modéliser quand même en PL des problèmes qui ne respectent pas a priori les conditions ci-dessus.

Deux grandes familles de méthodes de résolution de PL



Méthodes de points intérieur



Méthodes de simplexe

Résumé de ce qui a été vu

- Spécificités de la PL
- Modélisation mathématique
- Tracé, Interprétation et Résolution Graphique du cas 2D

Aperçu de la partie II

- Programmes linéaires à $n > 2$ variables : Résolution par Simplexe¹

1. pré-requis : calculs matriciels, résolution de systèmes linéaires par pivot de Gauss

Programmation Linéaire

II. $n > 2$ dimensions: résolution par la méthode du Simplexe

Sandra U. Ngueveu

Toulouse INP N7 - LAAS/CNRS
<https://homepages.laas.fr/sungueve>

<https://moodle-n7.inp-toulouse.fr/course/view.php?id=2380>

2A SN B-M-L : 2024/2025

Sandra U. Ngueveu (N7 - LAAS)

R.O. - support de prise de notes - 2A SN

2A SN B-M-L : 2024/2025

1 / 16

Principes sous-jacents

Résolution de PL de $n > 2$ variables par Simplexe

1 Principes sous-jacents

2 Méthode du simplexe (primal) et illustration

3 Pour aller plus loin

Systèmes de m équations à n inconnues (rappel)

Soit un système de m équations à n inconnues. Si les m équations sont linéairement indépendantes, alors on sait qu'il admet :

- soit 0 solution, en particulier lorsque $m > n$
- soit une seule solution ($\Leftrightarrow m = n$)
- soit une infinité de solutions ($\Leftrightarrow m < n$)

Algorithm 1 Pivot de Gauss pour résoudre un système de n équations à n inconnues

- 1: **POUR** $i = 1$ à n **FAIRE**
- 2: $pivot = a_{ii}$
- 3: Diviser tous les termes de la ligne i par le **pivot**
- 4: Mettre à 0 les termes a_{ki} de toutes les autres lignes $k \neq i$, par **combinaison linéaire**
- 5: **FIN POUR**

Application : résoudre le système linéaire suivant

$$\begin{cases} x_1 - 2x_2 + x_3 = 3 \\ 2x_1 + x_2 - x_3 = 7 \\ 3x_1 - x_2 + 2x_3 = 6 \end{cases}$$

PL dont la solution est triviale

$$\left. \begin{array}{l} (\text{PL1}) \quad \max 3x_1 + 2x_2 \\ \text{s.c.} \\ 2x_1 + x_2 + x_3 = 18 \\ 2x_1 + 3x_2 + x_4 = 42 \\ 3x_1 + x_2 + x_5 = 24 \\ x_1, x_2, x_3, x_4, x_5 \geq 0 \end{array} \right\} \Rightarrow \text{Solution Optimale Pas Evidente}$$

⇓ En fait il s'agit du même problème !

$$\left. \begin{array}{l} (\text{PL2}) \quad \max 33 - \frac{5}{4}x_3 - \frac{1}{4}x_4 \\ \text{s.c.} \\ x_1 + \frac{3}{4}x_3 - \frac{1}{4}x_4 = 3 \\ x_2 - \frac{1}{2}x_3 + \frac{1}{2}x_4 = 12 \\ x_5 - \frac{7}{4}x_3 + \frac{1}{4}x_4 = 3 \\ x_1, x_2, x_3, x_4, x_5 \geq 0 \end{array} \right\} \Rightarrow \text{Solution Optimale Triviale} \\ (\text{Laquelle?})$$

PL dont la solution est triviale

$$(PL1) \quad \begin{aligned} & \max 3x_1 + 2x_2 \\ & \text{s.c.} \\ & \begin{aligned} & 2x_1 + x_2 + x_3 = 18 \\ & 2x_1 + 3x_2 + x_4 = 42 \\ & 3x_1 + x_2 + x_5 = 24 \\ & x_1, x_2, x_3, x_4, x_5 \geq 0 \end{aligned} \end{aligned} \left. \right\} \Rightarrow \text{Solution Optimale Pas Evidente}$$

Comment passer de (PL1) à (PL2)

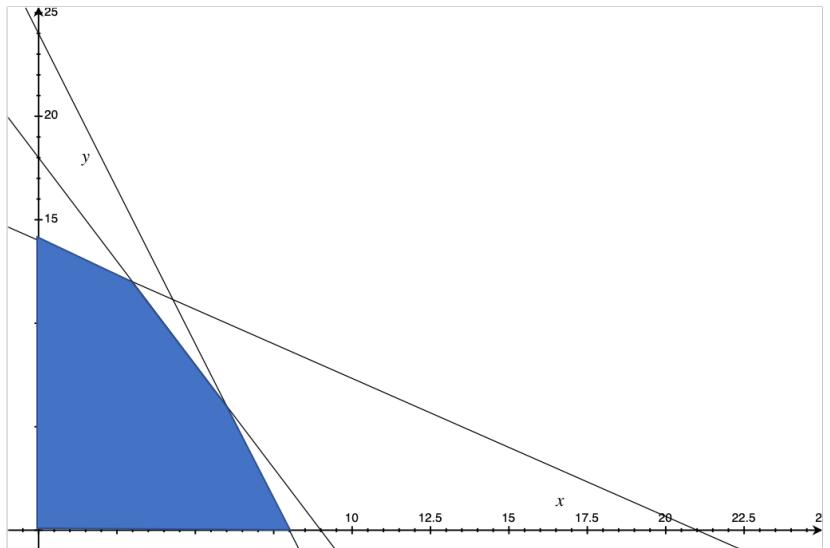
- ① Remplacer la 1ère contrainte par $\frac{3}{4}L_1 - \frac{1}{4}L_2$
- ② Remplacer la 2ème contrainte par $-\frac{1}{2}L_1 + \frac{1}{2}L_2$
- ③ Remplacer la 3ème contrainte par $-\frac{7}{4}L_1 + \frac{1}{4}L_2 + L_3$
- ④ Faire disparaître x_1 et x_2 de la fonction-objectif (en remplaçant par x_3 et x_4 en se servant des contraintes-égalités)

Lien entre forme standard et sommets de polyèdre

$$(PL) \quad \begin{aligned} & \max 3x_1 + 2x_2 \\ & \text{s.c.} \\ & \begin{aligned} & 2x_1 + x_2 \leq 18 \\ & 2x_1 + 3x_2 \leq 42 \\ & 3x_1 + x_2 \leq 24 \\ & x_1, x_2 \geq 0 \end{aligned} \end{aligned} \left. \right\} \Rightarrow$$



$$(PLS) \quad \begin{aligned} & \max 3x_1 + 2x_2 \\ & \text{s.c.} \\ & \begin{aligned} & 2x_1 + x_2 + x_3 = 18 \\ & 2x_1 + 3x_2 + x_4 = 42 \\ & 3x_1 + x_2 + x_5 = 24 \\ & x_1, x_2, x_3, x_4, x_5 \geq 0 \end{aligned} \end{aligned} \left. \right\} \Leftrightarrow$$



sommet de polyèdre $\Leftrightarrow n - m$ variables nulles pour la forme standard

Il est alors intéressant de se rappeler que la solution optimale correspond à un sommet du polyèdre de la forme canonique (cf résolution graphique 2D)

1 Principes sous-jacents

2 Méthode du simplexe (primal) et illustration

3 Pour aller plus loin

Concepts de base

Propriété 1

Si toutes les contraintes sont exprimées sous forme d'égalités, il y a problème d'optimisation si et seulement si $m < n$ et que les m équations sont linéairement indépendantes (voir slide 3).

Dans ce cas (le seul que nous considérerons désormais) :

- On peut choisir m variables et exprimer ces variables en fonction des $n - m$ vars restantes. (En faisant comme s'il s'agissait d'un système de m équations à m inconnues)
- Un choix de ces m variables est une **base** et les m variables sont appelées **variables de base**.
- Les $n - m$ vars restantes sont appelées **variables libres** ou variables hors base.
- Si les valeurs des vars libres sont connues, alors on peut en déduire celles des vars de base
- Les solutions où toutes les variables libres sont nulles sont appelées **solutions de base**.

Propriété 2

Si un programme linéaire admet une solution optimale, alors il existe une solution optimale qui est une solution de base.

Tirant partie de cette propriété, le simplexe ne se concentre que sur des solutions de base.

Concepts de base

Principe

Le simplexe consiste à se déplacer de base réalisable en base réalisable de telle sorte que chaque solution de base soit meilleure que la précédente jusqu'à trouver une solution optimale.

Pour cela il faut :

- connaitre une première solution de base réalisable
- savoir passer d'une base à une autre
- savoir reconnaître une solution de base optimale

La manipulation se fait à l'aide de tableaux :

$$\left. \begin{array}{l}
 \max 3x_1 + 2x_2 \\
 \text{s.c.} \\
 2x_1 + x_2 \leq 18 \\
 2x_1 + 3x_2 \leq 42 \\
 3x_1 + x_2 \leq 24 \\
 x_1, x_2 \geq 0 \\
 \text{modèle initial}
 \end{array} \right\} \Rightarrow \left. \begin{array}{l}
 \max 3x_1 + 2x_2 \\
 \text{s.c.} \\
 2x_1 + x_2 + x_3 = 18 \\
 2x_1 + 3x_2 + x_4 = 42 \\
 3x_1 + x_2 + x_5 = 24 \\
 x_1, x_2, x_3, x_4, x_5 \geq 0 \\
 \text{forme standard}
 \end{array} \right\} \Rightarrow \begin{array}{c|ccccc|c}
 & x_1 & x_2 & x_3 & x_4 & x_5 & b \\
 \hline
 x_3 & 2 & 1 & 1 & 0 & 0 & 18 \\
 x_4 & 2 & 3 & 0 & 1 & 0 & 42 \\
 x_5 & 3 & 1 & 0 & 0 & 1 & 24 \\
 \hline
 z & -3 & -2 & 0 & 0 & 0 & 0
 \end{array} \quad \text{tableau du simplexe}$$

Connaitre une première solution de base réalisable

Propriété 3

Dans un tableau du simplexe, les colonnes des variables de base, plus la colonne $\underbrace{(0, \dots, 0)}_m, 1^T$, doivent former une matrice identité $I^{\{m+1\}}$.

Les variables ajoutées lors de la transformation d'inéquations en équations sont souvent de bonnes candidates à la base initiale.

Remplissage du 1er tableau du simplex, à partir de la forme standard

- la ligne-objectif est remplie des coefs de la fonction-obj
- chaque colonne, sauf la dernière, correspond à une variable
- la dernière colonne contient les termes de droite (tjrs positifs)
- chaque ligne correspond à une contrainte et ne contient qu'une var de base (pour pouvoir former la matrice identité)
- on peut associer à chaque contrainte la var de base qu'elle contient

Savoir passer d'une base à une autre

Changer de base \Leftrightarrow échanger des vars de base avec des vars actuellement libres.

Simplexe opère 1 chgmt par itération : 1 var est sortie de la base et remplacée par 1 var libre.

Choix de la variable (libre) entrante x_e

- Seules sont candidates les vars dont les colonnes ont un terme **négatif** dans la ligne-objectif
(Pourquoi ?)
- Choix empirique : choisir la colonne j ayant le coef le plus **négatif** dans la ligne-objectif

Choix de la variable (de base) sortante x_s

- Pour garantir que la nouvelle base reste réalisable, seules sont candidates les vars de base ayant un coef positif dans la colonne de la var entrante déjà identifiée
- Règle du ratio : choisir la var de base (ligne) conduisant au plus petit ratio entre la colonne b et la colonne de la variable entrante

Application du changement de base

- Faire rentrer x_e et sortir x_s et faire ressortir la matrice identité pour les nvelles vars de base
- Exécution : appliquer des combinaisons linéaires entre la ligne de x_s et les autres lignes pour faire apparaître l'identité sur la nouvelle base

Savoir reconnaître une solution de base optimale

Propriété

Lors du choix de la variable entrante, s'il n'existe aucun terme **négatif** dans la ligne-objectif, alors la solution courante est optimale.

Lire le tableau final du simplexe

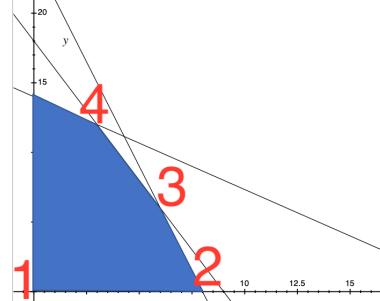
- valeur des variables de la solution optimale
- coût optimal

Application

Illustration 2D

Changements de base successifs au cours des itérations du Simplexe

$$(PLS) \quad \begin{aligned} & \max 3x_1 + 2x_2 \\ & \text{s.c.} \\ & \left. \begin{aligned} & 2x_1 + x_2 + x_3 = 18 \\ & 2x_1 + 3x_2 + x_4 = 42 \\ & 3x_1 + x_2 + x_5 = 24 \\ & x_1, x_2, x_3, x_4, x_5 \geq 0 \end{aligned} \right\} \end{aligned} \Rightarrow$$



	x_1	x_2	x_3	x_4	x_5	b
x_3	2	1	1	0	0	18
x_4	2	3	0	1	0	42
x_5	3	1	0	0	1	24
z	-3	-2	0	0	0	0

	x_1	x_2	x_3	x_4	x_5	b
x_2	0	1	3	0	-2	6
x_4	0	0	-7	1	4	12
x_1	1	0	-1	0	1	6
z	0	0	3	0	-1	30

	x_1	x_2	x_3	x_4	x_5	b
x_3	0	$\frac{1}{3}$	1	0	$-\frac{2}{3}$	2
x_4	0	$\frac{7}{3}$	0	1	$-\frac{2}{3}$	26
x_1	1	$\frac{1}{3}$	0	0	$\frac{1}{3}$	8
z	0	-1	0	0	1	24

	x_1	x_2	x_3	x_4	x_5	b
x_2	0	1	$-\frac{1}{2}$	$\frac{1}{2}$	0	12
x_5	0	0	$-\frac{7}{4}$	$\frac{1}{4}$	1	3
x_1	1	0	$\frac{3}{4}$	$-\frac{1}{4}$	0	3
z	0	0	$\frac{5}{4}$	$\frac{1}{4}$	0	33

1 Principes sous-jacents

2 Méthode du simplexe (primal) et illustration

3 Pour aller plus loin

Pour aller plus loin

- Que faire s'il n'existe pas de solution de base évidente pour le problème (P) ?
 - Introduire des variables artificielles
 - Méthode de pénalisation
 - Méthode à 2 phases
- Cas particuliers
 - Multiples solutions
 - Solution non bornée
 - Dégénérescence
- Méthode du simplexe dual
- Analyse de sensibilité en programmation linéaire

Résumé de ce qui a été vu

- Résolution de PL à $n \geq 2$ variables par la méthode du Simplexe
- Liens avec la méthode graphique

Programmation Linéaire en Nombres Entiers

I. Définition et principes généraux

Sandra U. Ngueveu

Toulouse INP N7 - LAAS/CNRS
<https://homepages.laas.fr/sungueve>

<https://moodle-n7.inp-toulouse.fr/course/view.php?id=2380>

2A SN B-M-L : 2024/2025

1 De la PL à la PLNE : Principales différences

2 Problèmes combinatoires

3 Aperçu de complexité

Différences entre les cas continus et discrets

Problème (P1) = Fabrication de ciment avec nouvelles durées

Problème (P2) = (P1) + nouvelles règles :

- les ciments A and B doivent être stockés dans des sacs séparés d'1kg
- chaque sac utilisé doit être plein à la fin de la journée

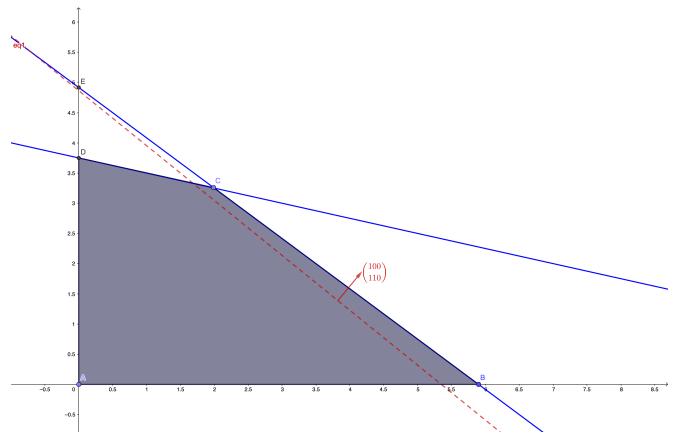
$$(P2) \max 100x_A + 110x_B \quad (1)$$

s.t.

$$100x_A + 120x_B \leq 590 \quad (2)$$

$$50x_A + 200x_B \leq 750 \quad (3)$$

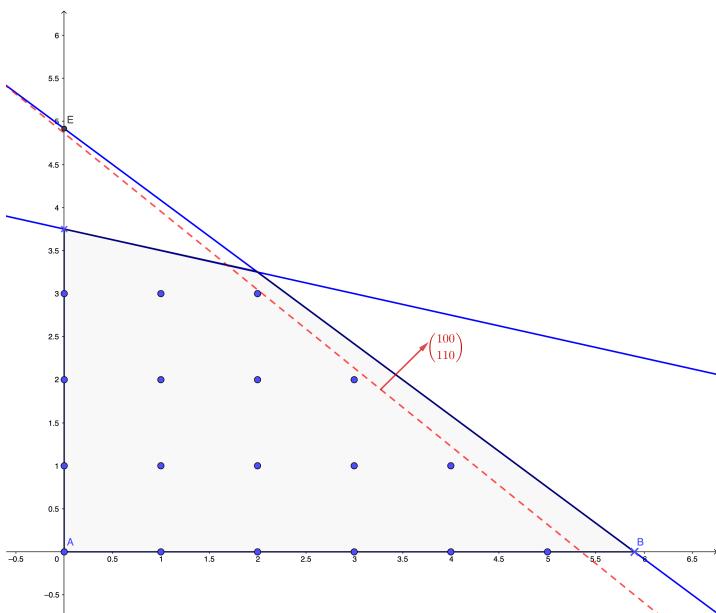
$$x_A, x_B \in \mathbb{N} \quad (4)$$



- La solution optimale de (P1) n'est pas réalisable pour (P2)
- La localisation de la solution optimale de (P2) n'est pas évidente

Problem (P2) : fabrication et stockage de ciment

Représentation graphique de (P2)



ensemble de solutions réalisables ?

polyèdre (convexe) ?

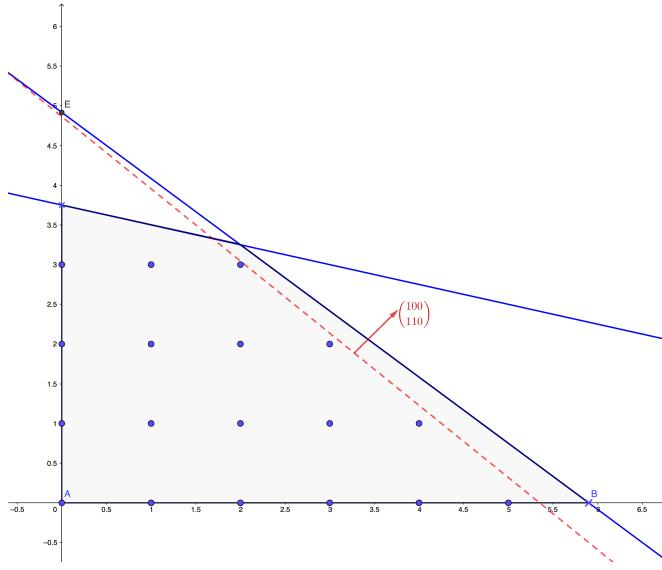
point extrémum vs points intérieurs ?

solution optimale ?

Arrondir les solutions de PL aux entiers les plus proches ?

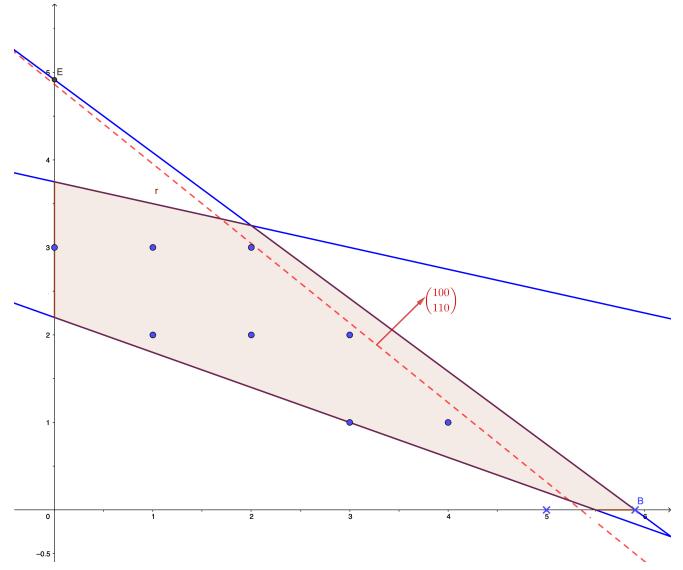
Pourquoi ne pas résoudre la relaxation linéaire PL et arrondir la solution fractionnaire pour obtenir la solution entière requise ?

l'optimalité ne serait pas garantie



Problème (P2)

la faisabilité ne serait pas garantie



Problème (P3) = (P2) $\cup \{x_1 + 5x_2 \geq 11\}$

1 De la PL à la PLNE : Principales différences

2 Problèmes combinatoires

3 Aperçu de complexité

Problèmes combinatoires

Soient :

- S un ensemble fini
- f une fonction qui attribue un coût $f(s)$ à chaque élément $s \in S$

Un problème combinatoire consiste à trouver l'élément s_0 de S qui minimise f .

Cela revient à résoudre le problème :

$$\min f(s) \quad (5)$$

sous les contraintes (s.c.)

$$s \in S \quad (6)$$

où S est un ensemble discret

Problèmes combinatoires

Cette définition est plus large qu'il n'y paraît au premier abord car elle englobe également :

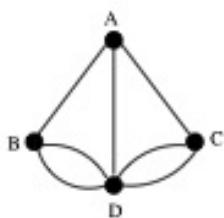
- les problèmes de maximisation, car $\min f \Leftrightarrow \max(-f)$
- les problèmes de calcul d'une valeur optimale
- les problèmes de décision ou d'existence dont on attend une réponse du type "oui ou "non"

Elle peut être adaptée au cas multi-objectif, stochastique, robustesse ou dynamique.

Pourquoi l'énumération est hors de question ?

L'ensemble des solutions réalisables est fini. Ne pourrait-on pas tout simplement tous les énumérer pour identifier la meilleure ?

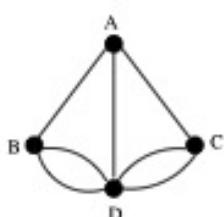
Le problème du voyageur de commerce



- Pour 3 sommets, combien de solutions réalisables ?
- Pour 5 sommets, combien de solutions réalisables ?
- Pour 20 sommets, combien de solutions réalisables ?
- Combien de temps pour trouver la meilleure solution pour 20 sommets avec un processeur capable d'évaluer 1 millions de solutions par secondes ?
- Combien de processeurs en parallèle faudrait-il pour garder un temps de calcul similaire si un noeud supplémentaire est ajouté ? deux noeuds supplémentaires ?

Explosion combinatoire

Explosion combinatoire : $O(n!)$ solutions



20 noeuds $\approx 1e^{17}$ solutions

- Proc. 3GHz : 3 op / nano seconde
- Proc. Planck : 1 op / temps de Planck (5.39×10^{-44})s
- //P. Planck : remplir l'univers avec des processeurs de Planck d'1mm³

Noeuds	Proc. 3 GHz	Proc. Planck	//P. Planck

⇒ Analyse théorique de classes de problèmes pour en réduire l'espace de recherche

1 De la PL à la PLNE : Principales différences

2 Problèmes combinatoires

3 Aperçu de complexité

Les outils de l'optimisation combinatoire sont empruntés de :

- Programmation linéaire en nombre entiers ou mixte
- Programmation par contraintes (I.A.)
- Théorie des graphes
- Simulation
- Algorithmique + complexité

Certains calculs peuvent être faits par des outils commerciaux d'optimisation, mais bien souvent il faut concevoir un algorithme car **il n'existe pas de méthode générique performante quelque soit le problème ou l'instance en optimisation combinatoire.**

Evaluation d'un algorithme

Comment évaluer la qualité d'un algorithme ou comparer 2 algorithmes ?

Deux critères généralement importants sont :

- le temps de calculs
- l'espace mémoire utilisé

D'autres critères, comme la difficulté d'implémentation, la fiabilité, ... peuvent rentrer en compte.

Certains résultats peuvent fortement dépendre des instances résolues et des données utilisées. Mais en général, ce que l'on veut mesurer en priorité c'est si l'algorithme gardera, même dans les **pires cas**, des temps de calcul raisonnables.

Evaluation d'un algorithme

Complexité¹

La complexité est une fonction de la taille des données pour prédire l'ordre de grandeur de la variation des temps de calculs quand on passe de petits jeux d'essai de grands problèmes.

Cela permet aussi d'estimer la taille maximale des problèmes qui pourront être résolus en pratique.

La complexité d'un algorithme A travaillant sur des données est une fonction qui exprime le nombre d'instructions exécutées :

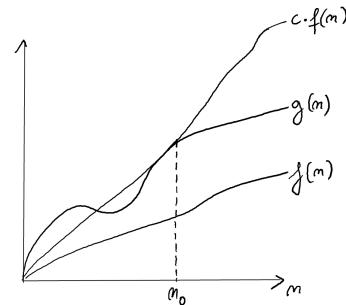
- à l'ordre près (notation O)
- dans le pire des cas,
- en fonction de la taille n des données.

1. rien à voir avec la difficulté de programmation !!

La notation O

Pour une fonction f croissante définie sur l'ensemble des entiers positifs, $O(f)$ est la classe des fonctions g définies sur l'ensemble des entiers positifs telles que : $\exists c > 0, \exists n_0 > 0$ tels que $\forall n \geq n_0, g(n) \leq c \cdot f(n)$.

- La figure montre une fonction g qui vérifie $g \in O(f)$: à partir de n_0 , la courbe $g(n)$ est comprise entre les courbes de $f(n)$ et $c \cdot f(n)$
- $O(1)$ est l'ensemble des fonctions majorées par une constante à partir d'un certain rang ; par exemple $g(n) = \frac{1}{n} \in O(1)$.



Le comportement asymptotique des fonctions étant étudié à une constante multiplicative positive près, de nombreuses classes de fonctions sont équivalentes.

- par exemple $O(e^n) = O(e^{n+173})$
- pour tout polynôme $P(n) = a_p n^p + a_{p-1} n^{p-1} + \dots + a_1 n + a_0$, on a $P \in O(n^p)$.
 - ainsi $O(12n^5 - 4n^3 + 2n^2 + n - 1) = O(n^5)$.

Evaluation d'un algorithme

Il y a donc deux types d'algorithmes, en fonction de leur complexité :

- ceux dits **polynomiaux**
 - complexité de l'ordre d'un polynôme
 - ex : $\log n, n^{0.5}, n \log n, n^2, O(n^{\log n}), O(n^{2.5}), \dots$
- ceux dits **exponentiels**
 - vraie exponentielle au sens mathématique $e, 2^n, \dots$
 - fonction comme factorielle $n!$, ou n^n, k^n, n^n, \dots

Explosion Combinatoire

Taille-> Complexité	20	50	100	200	500	1000
$10^3 n$	0.02 s	0.05 s	0.1 s	0.2 s	0.5 s	1 s
$10^3 n \log_2 n$	0.09 s	0.3 s	0.6 s	1.5 s	4.5 s	10 s
$100 n^2$	0.04 s	0.25 s	1 s	4 s	25 s	2 min
$10 n^3$	0.02 s	1 s	10 s	1 min	21 min	27 h
$n^{\log_2 n}$	0.4 s	1.1 h	220 j	12500 ans	$5 \cdot 10^{10}$ ans	--
$2^{n/3}$	0.0001 s	0.1 s	2.7 h	$3 \cdot 10^6$ ans	--	--
2^n	1 s	36 ans	--	--	--	--
3^n	58 min	$2 \cdot 10^{11}$ ans	--	--	--	--
$n!$	77100 ans	--	--	--	--	--

Figure – Croissance du temps de calcul pour différentes complexités

Les cases non remplies ont des durées supérieures à 1000 miliards d'années.

Complexité d'un problème / Complexité d'un algorithme

Les problèmes d'Optimisation Combinatoire pour lesquels vérifier la validité d'une solution est polynomial, se répartissent en deux grandes classes :

- ceux qui sont résolus de manière optimale par des algorithmes **polynomiaux** (classe P)
- ceux dont la résolution optimale peut prendre un temps **exponentiel** dans le pire cas (classe NP)

La notion de problème **NP-complet/NP-difficile** permet de mieux comprendre pourquoi certains problèmes ne disposent toujours pas d'algorithmes efficaces à l'heure actuelle.

La complexité d'un problème corresponds à la complexité du **meilleur algorithme capable de le résoudre**.

Complexité de problèmes combinatoires

Exemple de problème de complexité linéaire

Rechercher un élément dans une liste

...

Exemples de problèmes de complexité polynomiale

Problèmes de programmation linéaire

Problèmes d'affectation / couplage maximal

Plus courts chemins

...

Exemples de problèmes de complexité exponentielle

Voyageur de Commerce

Job Shop

Coloration des sommets d'un graphe

...

Exemple classique

Il suffit parfois d'ajouter une contrainte/variable/donnée/variante pour qu'un problème passe de *polynomial* à *NP-difficile*.

Le pb d'affectation

Soient n produits et n machines. Fabriquer i coûte d_{ij} si i est affecté au à la machine j .
On veut affecter les produits aux machines pour tout traiter en minimisant le coût total.

Le pb d'affectation généralisée

Soient m produits, n machines. Fabriquer i dure a_{ij} et coûte d_{ij} si i est fabriqué par la machine j . La durée max de travail d'une machine j est b_j .
On veut affecter les produits aux machines pour tout traiter en minimisant le coût total.

... MODELE MATH ... ?

... MODELE MATH ... ?

POLYNOMIAL !!!

NP-DIFFICILE !!!

Comment résoudre un problème NP-difficile ?

- énumération complète (si très petits problèmes)
- énumération intelligente (problèmes de taille moyenne ou grande)
 - programmation dynamique (résolution de sous-problèmes emboités),
 - méthodes arborescentes (séparation en sous-cas dont beaucoup sont éliminés par des tests).
 - ...
- méthodes approchées (problèmes de grande ou très grande taille)
bonnes solutions mais sans garantie d'optimalité.
 - (méta-)heuristiques
 - recherches locales
 - ...

Résumé de ce qui a été vu

- Définition de l'optimisation combinatoire
- Relation et différence avec l'optimisation continue
- Problème vs instance : besoin d'algorithmes
- Comment évaluer un algorithme
- Explosion combinatoire : pourquoi un PC plus puissant ne résouds rien

Programmation Linéaire en Nombres Entiers

II. Branch-and-Bound (Procédures de séparation et évaluation)

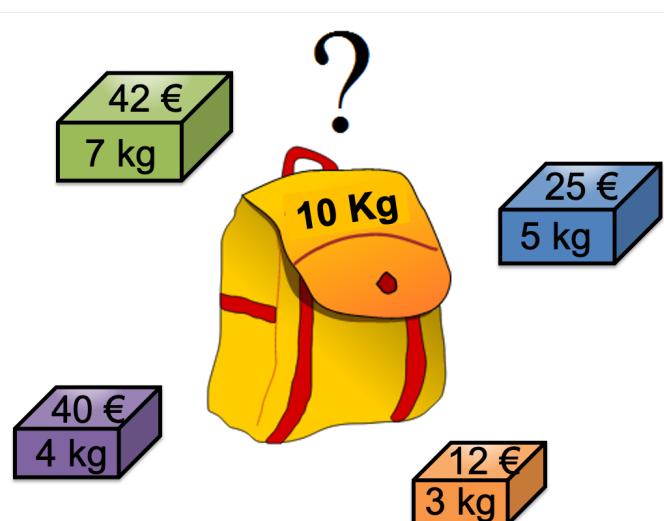
Sandra U. Ngueveu

Toulouse INP N7 - LAAS/CNRS
<https://homepages.laas.fr/sungueve>

<https://moodle-n7.inp-toulouse.fr/course/view.php?id=2380>

2A SN B-M-L : 2024/2025

Cas illustratif : le problème du sac à dos (PS)



Données :

- sac à dos de capacité Q ,
- $n = 4$ objets disponibles,
- l'objet i a un poids w_i et un coût c_i

Quels objets choisir pour maximiser le coût total en respectant la capacité du sac à dos ?

Variables

?

Fonction-objectif

?

Contraintes

?

Domaine

?

1 Relaxations et calculs de bornes

2 Arbre de décision / Arborescence

3 Branch-and-bound

4 Règles à respecter pour un algorithme correct

5 Exemples d'applications

Relaxation d'un PLNE

Soient deux problèmes (P) et (RP) , soient $S_{(P)}$ et $S_{(RP)}$ les ensembles de solutions de ces problèmes, et soient $s_{(P)}^*$ et $s_{(RP)}^*$ leurs solutions optimales.

- Relaxation

(RP) est relaxation de (P) si et seulement si $S_{(P)} \subset S_{(RP)}$

- Relaxation linéaire

(RP) est relaxation linéaire de (P) si et seulement si
 $(P) = (RP) \cup \{\text{contraintes d'intégralité}\}$

- Autres types de relaxations

suppression ou agrégation de contraintes
relaxation Lagrangienne

...

NB1 : Si (P) est un problème de minimisation, alors $f(s_{(RP)}^*) \leq f(s_{(P)}^*)$.

NB2 : Si la solution optimale de (RP) est réalisable pour (P) , alors elle est une solution optimale de (P) .

Bornes Inférieure(BI) Supérieure(BS) Primale(BP) Duale(BD)

Définitions

- BS = toute valeur supérieure ou égale à la valeur optimale
- BI = toute valeur inférieure ou égale à la valeur optimale
- Le coût de toute solution réalisable fournit une BP
- Toute solution optimale d'une relaxation fournit une BD

Comment obtenir des bornes BI et BS pour un problème d'optimisation ?

- En **minimisation** : BI = BD et BS = BP
- En **maximisation** : BI = BP et BS = BD

Intérêt de BP : la solution correspondante est réalisable

Intérêts de BD : évaluer

- la qualité d'une solution réalisable, en prouver l'optimalité
- l'écart à l'optimum alors qu'on ne connaît pas la solution optimale
- s'il est pertinent d'allouer plus de temps de calcul à la recherche d'une meilleure solution

Exemples de calcul de borne pour le problème du sac à dos

Borne 1 = capacité $\times \max_i r_i$ où $r_i = \frac{c_i}{w_i}$

- Donner la relaxation de (PS) dont ce calcul est la solution optimale, pour prouver qu'il s'agit bien d'un calcul de borne supérieure valide
- Appliquer l'algorithme pour obtenir une borne supérieure de (PS)
- Que peut-on en déduire ?

Exemple de calcul de borne pour le problème du sac à dos

Borne 2 : résoudre la relaxation linéaire (appelée problème du sac à dos fractionnaire - PSF)

$$(PSF) \max 42x_1 + 40x_2 + 12x_3 + 25x_4 \quad (1)$$

S.C.

$$7x_1 + 4x_2 + 3x_3 + 5x_4 \leq 10 \quad (2)$$

$$0 \leq x_i \leq 1, \forall i \in \{1, 2, 3, 4\} \quad (3)$$

(PSF) peut être résolu par un algorithme de programmation linéaire tel que simplex. Mais il peut aussi être résolu par un algorithme polynomial basé sur le tri de tous les objets par ordre décroissant de ratio r_i

- Appliquer l'algorithme pour obtenir une borne supérieure de (PS)
- Que peut-on en déduire ?

1 Relaxations et calculs de bornes

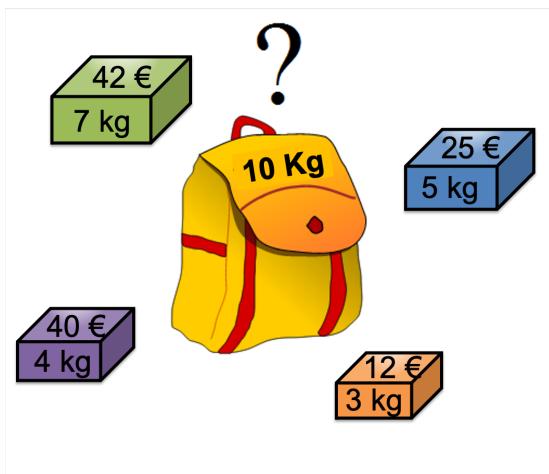
2 Arbre de décision / Arborescence

3 Branch-and-bound

4 Règles à respecter pour un algorithme correct

5 Exemples d'applications

Arbre de décision / Arbre des sous-problèmes



$$\max 42x_1 + 40x_2 + 12x_3 + 25x_4 \quad (4)$$

S.C.

$$7x_1 + 4x_2 + 3x_3 + 5x_4 \leq 10 \quad (5)$$

$$x_1, x_2, x_3, x_4 \in \{0, 1\} \quad (6)$$

- Construire un arbre de décision associé au problème
- En déduire l'arbre des sous-problèmes associés

L'arbre a 2^n feuilles (\Rightarrow explosion combinatoire)

MAIS si nous pouvions identifier quels sous-arbres contiennent la solution optimale, alors nous pourrions réduire drastiquement l'espace de recherche

1 Relaxations et calculs de bornes

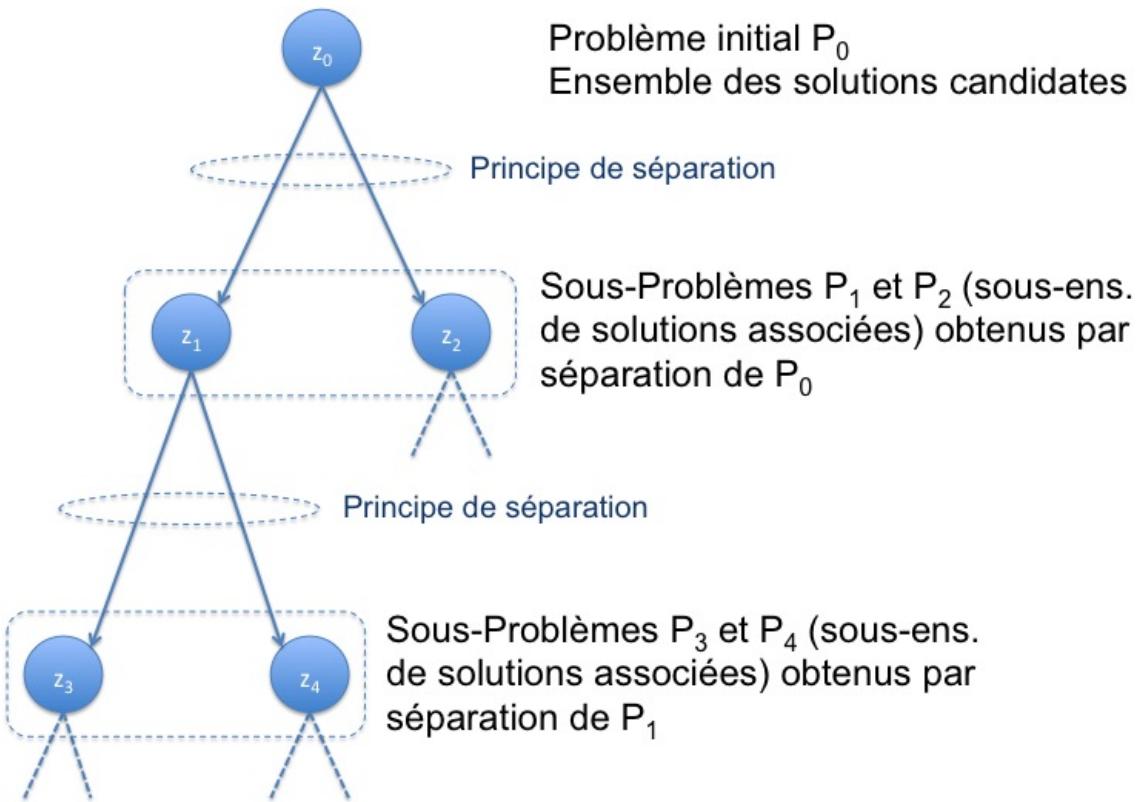
2 Arbre de décision / Arborescence

3 Branch-and-bound

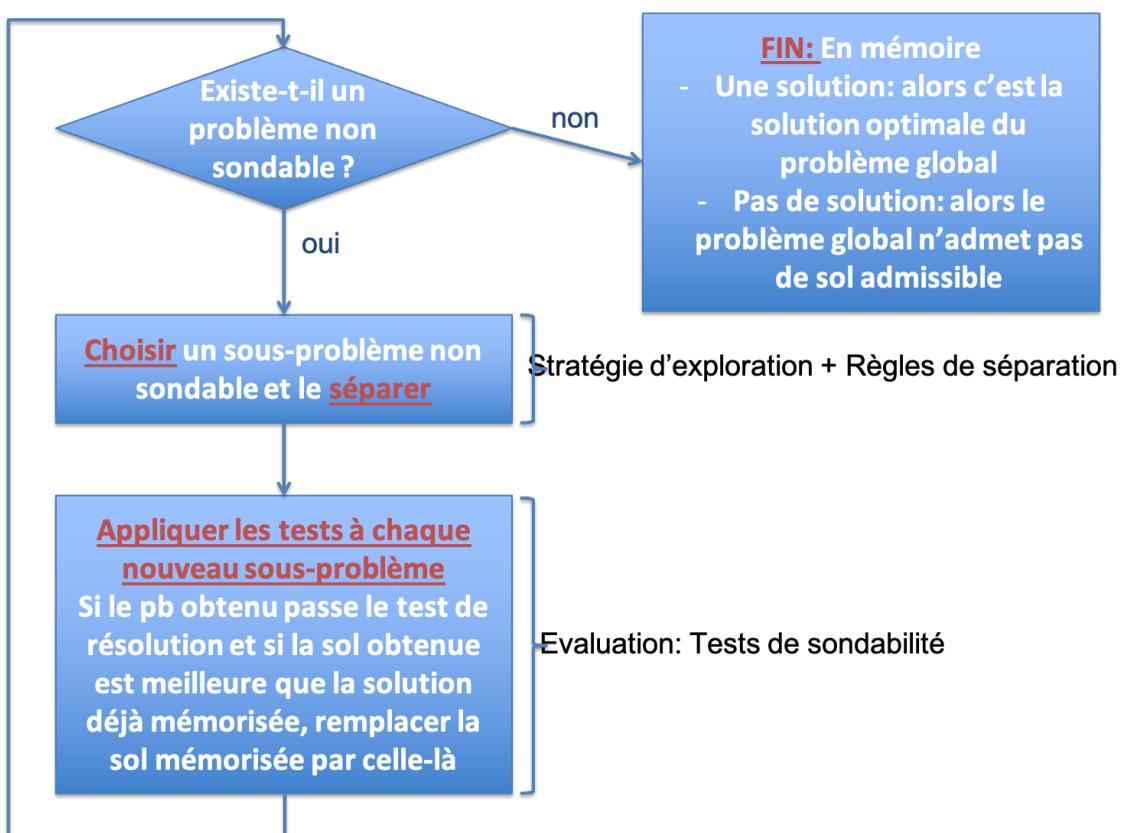
4 Règles à respecter pour un algorithme correct

5 Exemples d'applications

Branch-and-Bound pour PLNE



Branch-and-Bound pour PLNE



Branch-and-Bound pour PLNE

Principe :

- Séparer progressivement le problème en sous-problèmes traitables (aussi appelés "sondables")

Objectif

- Enumérer intelligemment l'espace des solutions
- Hiérarchiser les sous-problèmes sous forme d'arbre
- "tuer" des branches/nœuds au plus tôt lors de l'exploration de l'arbre

3 composantes :

- Règle de Séparation
- Tests de Sondabilité (souvent basés sur des bornes)
 - Test **Admissibilité**, Test **Optimalité**, Test **Résolution**
- Stratégie d'exploration

Exemple de B&B pour résoudre le problème du sac à dos

$$(PS) \max 2x_1 + 10x_2 + 8x_3 + 6x_4 \quad (7)$$

S.C.

$$2x_1 + 6x_2 + 5x_3 + 5x_4 \leq 10 \quad (8)$$

$$x_1, x_2, x_3, x_4 \in \{0, 1\} \quad (9)$$

1 Tests de sondabilité : basés sur les bornes supérieures (BS)

- Test d'**Admissibilité** : réussi si la capacité restante est strictement négative
- Test d'**Optimalité** : réussi si BS est pire que la meilleure solution connue de (PS)
- Test de **Résolution** : réussi si la solution fournissant BS est réalisable pour (PS)

(i) BS = Borne 1 du slide 7

(ii) BS = Borne 1 améliorée : supprimer les objets vérifiant poids > capacité avant d'appliquer Borne 1

2 Règle de séparation

- choisir la variable x_k dont la valeur n'est pas binaire lors du calcul de BS
- 2 sous-problèmes : $x_k = 1$ et $x_k = 0$

3 Stratégie d'exploration

- priorité au noeud ayant la plus grande BS
- priorité au noeud le plus à gauche (profondeur d'abord)

Arborescence 1 : Borne 1 et priorité à BS

$$P_0: \begin{cases} \text{Max } 2x_1 + 10x_2 + 8x_3 + 6x_4 \\ \text{s.c. } 2x_1 + 6x_2 + 5x_3 + 5x_4 \leq 10 \\ x_i \in \{0,1\}, \forall i \in \{1, \dots, 4\} \end{cases}$$

$$P_1: \begin{cases} \text{Max } 2x_1 + 8x_3 + 6x_4 + 10 \\ \text{s.c. } 2x_1 + 5x_3 + 5x_4 \leq 4 \\ x_i \in \{0,1\}, \forall i \in \{1,3,4\} \end{cases}$$

$$P_3: \begin{cases} \text{Max } 2x_1 + 6x_4 + 18 \\ \text{s.c. } 2x_1 + 5x_4 \leq -1 \\ x_i \in \{0,1\}, \forall i \in \{1,4\} \end{cases}$$

$$P_5: \begin{cases} \text{Max } 2x_1 + 6x_4 + 8 \\ \text{s.c. } 2x_1 + 5x_4 \leq 5 \\ x_i \in \{0,1\}, \forall i \in \{1,4\} \end{cases}$$

$$P_7: \begin{cases} \text{Max } 2x_1 + 16 \\ \text{s.c. } 2x_1 \leq -1 \\ x_i \in \{0,1\}, \forall i \in \{1\} \end{cases}$$

$$P_2: \begin{cases} \text{Max } 2x_1 + 8x_3 + 6x_4 \\ \text{s.c. } 2x_1 + 5x_3 + 5x_4 \leq 10 \\ x_i \in \{0,1\}, \forall i \in \{1,3,4\} \end{cases}$$

$$P_4: \begin{cases} \text{Max } 2x_1 + 6x_4 + 10 \\ \text{s.c. } 2x_1 + 5x_4 \leq 4 \\ x_i \in \{0,1\}, \forall i \in \{1,4\} \end{cases}$$

$$P_6: \begin{cases} \text{Max } 2x_1 + 6x_4 \\ \text{s.c. } 2x_1 + 5x_4 \leq 10 \\ x_i \in \{0,1\}, \forall i \in \{1,4\} \end{cases}$$

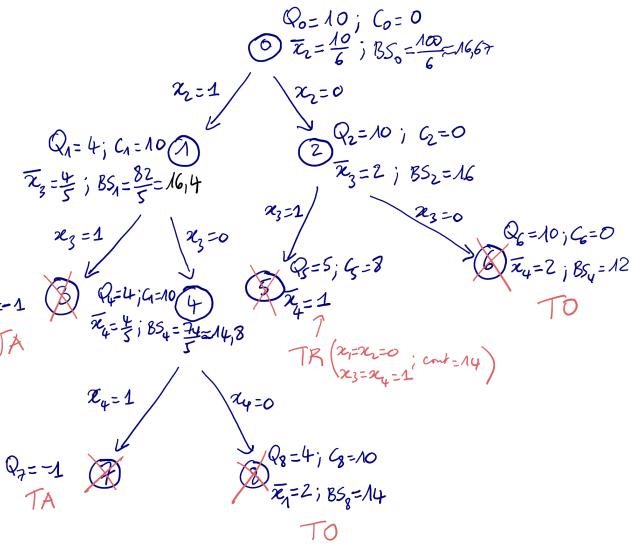
$$P_8: \begin{cases} \text{Max } 2x_1 + 10 \\ \text{s.c. } 2x_1 \leq 4 \\ x_i \in \{0,1\}, \forall i \in \{1\} \end{cases}$$

Q_i = capacité restante, C_i = coût des objets imposés dans le sac, BS_i = valeur de la borne au noeud i , \bar{x}_k = qté de l'objet k dans la solution BS

Mémoire

Sol: (✓) ($x_1=x_2=0; x_3=x_4=1$)
Cont: (✗) (Cont = 14)

cont	2	10	8	6
poids	2	6	5	5
ratio	1	1,66	1,60	1,12



File

- (✓)
- (✗)
- (✗)
- (✗)
- (✗)
- (✗)
- (✗)
- (✗)

Arborescence obtenue pour:
• BS = Borne 1
• Stratégie d'exploration : priorité au noeud de meilleure BS

Arborescence 2 : Borne 1 et priorité à profondeur

$$P_0: \begin{cases} \text{Max } 2x_1 + 10x_2 + 8x_3 + 6x_4 \\ \text{s.c. } 2x_1 + 6x_2 + 5x_3 + 5x_4 \leq 10 \\ x_i \in \{0,1\}, \forall i \in \{1, \dots, 4\} \end{cases}$$

$$P_1: \begin{cases} \text{Max } 2x_1 + 8x_3 + 6x_4 + 10 \\ \text{s.c. } 2x_1 + 5x_3 + 5x_4 \leq 4 \\ x_i \in \{0,1\}, \forall i \in \{1,3,4\} \end{cases}$$

$$P_3: \begin{cases} \text{Max } 2x_1 + 6x_4 + 18 \\ \text{s.c. } 2x_1 + 5x_4 \leq -1 \\ x_i \in \{0,1\}, \forall i \in \{1,4\} \end{cases}$$

$$P_5: \begin{cases} \text{Max } 2x_1 + 10 \\ \text{s.c. } 2x_1 \leq 4 \\ x_i \in \{0,1\}, \forall i \in \{1\} \end{cases}$$

$$P_7: \begin{cases} \text{Max } \cancel{\quad} \\ \text{s.c. } \cancel{\quad} \end{cases}$$

$$P_9: \begin{cases} \text{Max } 2x_1 + 6x_4 + 8 \\ \text{s.c. } 2x_1 + 5x_4 \leq 5 \\ x_i \in \{0,1\}, \forall i \in \{1,4\} \end{cases}$$

$$P_2: \begin{cases} \text{Max } 2x_1 + 6x_4 + 18 \\ \text{s.c. } 2x_1 + 5x_4 \leq -1 \\ x_i \in \{0,1\}, \forall i \in \{1,4\} \end{cases}$$

$$P_4: \begin{cases} \text{Max } 2x_1 + 16 \\ \text{s.c. } 2x_1 \leq -1 \\ x_i \in \{0,1\}, \forall i \in \{1\} \end{cases}$$

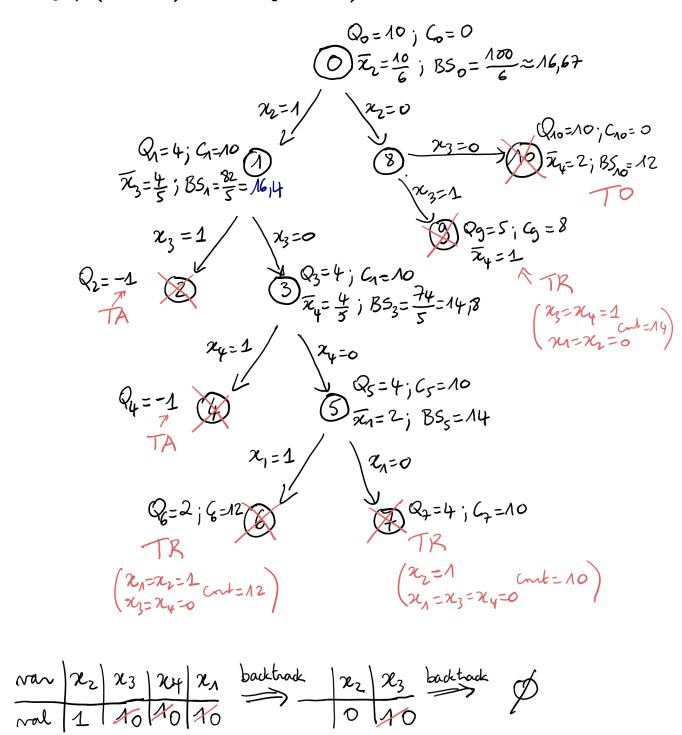
$$P_6: \begin{cases} \text{Max } \cancel{\quad} \\ \text{s.c. } \cancel{\quad} \end{cases}$$

$$P_8: \begin{cases} \text{Max } 2x_1 + 8x_3 + 6x_4 \\ \text{s.c. } 2x_1 + 5x_3 + 5x_4 \leq 10 \\ x_i \in \{0,1\}, \forall i \in \{1,3,4\} \end{cases}$$

$$P_{10}: \begin{cases} \text{Max } 2x_1 + 6x_4 \\ \text{s.c. } 2x_1 + 5x_4 \leq 10 \\ x_i \in \{0,1\}, \forall i \in \{1,4\} \end{cases}$$

Mémoire

Sol: (✗) ($x_1=x_2=1, x_3=x_4=0$) ($x_1=x_2=0, x_3=x_4=1$)
Cont: (✗) (Cont = 12)



Arborescence 3 : Borne 1 améliorée et priorité à BS

$$P_0: \begin{cases} \text{Max } 2x_1 + 10x_2 + 8x_3 + 6x_4 \\ \text{S.c. } 2x_1 + 6x_2 + 5x_3 + 5x_4 \leq 10 \\ x_i \in \{0,1\}, \forall i \in \{1, \dots, 4\} \end{cases}$$

$$P_1: \begin{cases} \text{Max } 2x_1 + 8x_3 + 6x_4 + 10 \\ \text{S.c. } 2x_1 + 5x_3 + 5x_4 \leq 4 \\ x_i \in \{0,1\}, \forall i \in \{1,3,4\} \end{cases} \Rightarrow \begin{cases} \text{Max } 2x_1 + 10 \\ \text{S.c. } 2x_1 \leq 4 \\ x_i \in \{0,1\}, \forall i \in \{1\} \end{cases}$$

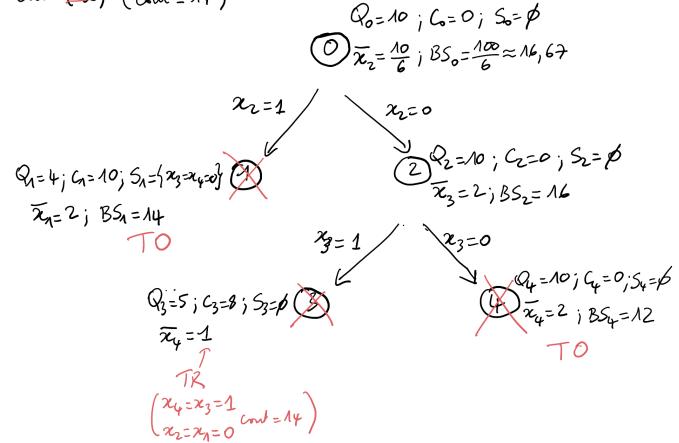
$$P_2: \begin{cases} \text{Max } 2x_1 + 8x_3 + 6x_4 \\ \text{S.c. } 2x_1 + 5x_3 + 5x_4 \leq 10 \\ x_i \in \{0,1\}, \forall i \in \{1,3,4\} \end{cases}$$

$$P_3: \begin{cases} \text{Max } 2x_1 + 6x_4 + 8 \\ \text{S.c. } 2x_1 + 5x_4 \leq 5 \\ x_i \in \{0,1\}, \forall i \in \{1,4\} \end{cases}$$

$$P_4: \begin{cases} \text{Max } 2x_1 + 6x_4 \\ \text{S.c. } 2x_1 + 5x_4 \leq 10 \\ x_i \in \{0,1\}, \forall i \in \{1,4\} \end{cases}$$

Mémoire

Sol: (1) ($x_4 = x_3 = 1; x_2 = x_1 = 0$)
 Cont: (14) (cont = 14)



File

(1) 16,67
 (2) 14
 (3) 16

Arborescence obtenue pour :

- BS = Boîne 1 améliorée
- Stratégie d'exploration : priorité au nœud de meilleure BS

Arborescence 4 : Borne 1 améliorée et priorité à profondeur

$$P_0: \begin{cases} \text{Max } 2x_1 + 10x_2 + 8x_3 + 6x_4 \\ \text{S.c. } 2x_1 + 6x_2 + 5x_3 + 5x_4 \leq 10 \\ x_i \in \{0,1\}, \forall i \in \{1, \dots, 4\} \end{cases}$$

$$P_1: \begin{cases} \text{Max } 2x_1 + 8x_3 + 6x_4 + 10 \\ \text{S.c. } 2x_1 + 5x_3 + 5x_4 \leq 4 \\ x_i \in \{0,1\}, \forall i \in \{1,3,4\} \end{cases} \Rightarrow \begin{cases} \text{Max } 2x_1 + 10 \\ \text{S.c. } 2x_1 \leq 4 \\ x_i \in \{0,1\}, \forall i \in \{1\} \end{cases}$$

$$P_2: \begin{cases} \text{Max } \\ \text{---} \\ \text{---} \end{cases}$$

$$P_3: \begin{cases} \text{Max } \\ \text{---} \\ \text{---} \end{cases}$$

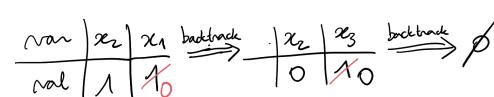
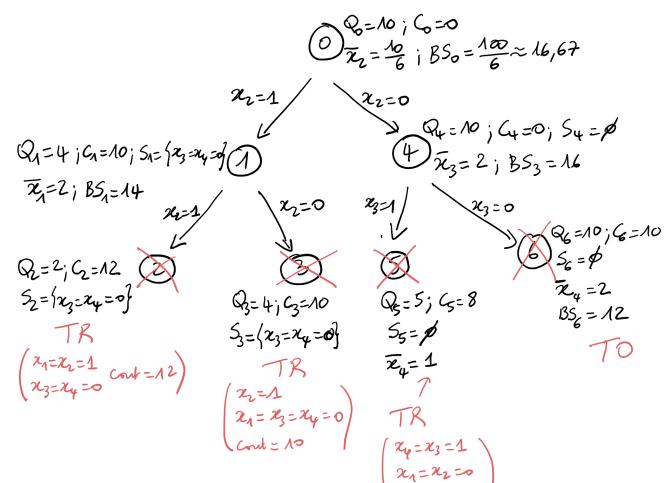
$$P_4: \begin{cases} \text{Max } 2x_1 + 8x_3 + 6x_4 \\ \text{S.c. } 2x_1 + 5x_3 + 5x_4 \leq 10 \\ x_i \in \{0,1\}, \forall i \in \{1, \dots, 4\} \end{cases}$$

$$P_5: \begin{cases} \text{Max } 2x_1 + 6x_4 + 8 \\ \text{S.c. } 2x_1 + 5x_4 \leq 5 \\ x_i \in \{0,1\}, \forall i \in \{1,4\} \end{cases}$$

$$P_6: \begin{cases} \text{Max } 2x_1 + 6x_4 + 10 \\ \text{S.c. } 2x_1 + 5x_4 \leq 10 \\ x_i \in \{0,1\}, \forall i \in \{1,4\} \end{cases}$$

Mémoire

Sol: (1) ($x_1 = x_2 = 1; x_3 = x_4 = 0$) ($x_1 = x_2 = 0; x_3 = x_4 = 1$)
 Cont: (12) (cont = 12) (cont = 14)



1 Relaxations et calculs de bornes

2 Arbre de décision / Arborescence

3 *Branch-and-bound*

4 Règles à respecter pour un algorithme correct

5 Exemples d'applications

Principe de Séparation

- Aucune solution ne doit être perdue ou ajoutée
 - l'union des ss-pbs reforme le pb de départ
- Souvent un **choix heuristique** validé par des tests numériques
 - choix glouton (ex : le plus petit arc libre pour PVC)
 - choix le plus informant (ex : la var présente dans le plus grande nbre de contraintes)
 - ...
- Le but est :
 - soit de précipiter la découverte d'une **nouvelle solution**
 - soit de détecter au plus vite des contradictions prouvant qu'un **nœud est vide**
- Une **séparation poussée à l'extrême** revient à fragmenter en ss-ens à une solution, donc **nombre exponentiel de nœuds explorés** (énumération complète). Pour éviter cela, on évite de séparer le nœud-père en "trop" de nœuds fils et on compte sur les tests **TA, TO, TR** pour réduire l'arborescence en identifiant au plus tôt les nœuds qui ne valent pas la peine d'être séparés à nouveau.

Par convention, les nœuds sont souvent numérotés dans l'ordre de leur création.

Stratégie d'Exploration

Il s'agit de définir l'ordre suivant lequel les nœuds seront étudiés

PSEP (procédure de séparation et évaluation progressive)

- Priorité au nœud qui donne la meilleure borne (BS en maximisation, BI en minimisation)
- Avantage : Les sol réalisables trouvées sont tt de suite de très bonne qualité
- Inconvénient : Requiert plus de mémoire (liste des nœuds non séparés)

PSES (procédure de séparation et évaluation séquentielle)

- Priorité toujours "à gauche" (1er des noeuds-fils nouvellement créés)
- Avantage : Requiert moins d'espace mémoire, Trouve plus vite une solution réalisable
- Inconvénient : Globalement moins rapide que PSEP

Autres

- Probabilité sur le choix de la prochaine branche à explorer
- Exploration en profondeur ("depth-first search")
- ...

Les tests numériques et les contraintes technologiques (espace mémoire, ...) permettent de valider/invalider le choix d'une stratégie d'exploration par rapport à une autre.

Tests de Sondabilité

Un nœud est sondable si on peut répondre "OUI" à un des 3 tests suivants :

TA : Test d'admissibilité = montrer que (mq) il n'y a pas de solution admissible

- ex1 : si contrainte du type $\geq b_i$, alors évaluer un majorant G_i^k du membre de gauche. Si $G_i^k < b_i$ alors même dans le meilleur des cas on ne pourra pas valider cette contrainte, donc inutile de continuer sur cette branche, "tuer" le nœud correspondant
- ex2 : raisonnement inverse si contrainte du type $\leq b_i$
- ex3 : mq une relaxation du problème est déjà infaisable

TO : Test d'optimalité = mq la sol optimale ne peut pas être au sein de ce nœud

- mq les coûts des sols du nœud seront toujours pires que celui d'une solution déjà connue
- ex : calcul d'une borne duale à comparer avec le coût d'une solution déjà connue (préalablement obtenue par le même B&B ou un autre algorithme)

TR : Test de résolution = mq le sous-problème se résoud de manière triviale

- sous-problème à une seule variable
- sous-problème polynomial

Tests de Sondabilité

Actions

- Un nœud sondable peut être éliminé.
- Un nœud non sondable doit être séparé en plusieurs ss-pbs suivant la **règle de séparation** prévue.

Un *branch-and-bound* efficace trouve un compromis entre la qualité des résultats fournis par les tests et l'investissement accepté en temps et volumes de calculs, car ces calculs doivent être faits pour chaque nœud exploré.

Branch-and-bound classique pour PLNE généraux

① Règle de séparation

- choisir une variable x_k ayant la valeur la plus fractionnaire notée v^* dans la solution optimale de la relaxation
- 2 sous-problèmes : $x_k \leq \lceil v^* \rceil$ et $x_k \geq \lfloor v^* \rfloor$

② Tests de sondabilité : basés sur la résolution de la relaxation linéaire (RL) du noeud courant pour obtenir des bornes

- Test d'**Admissibilité** : réussi si (RL) est infaisable
- Test d'**Optimalité** : réussi si la solution de (RL) est pire que la meilleure solution connue
- Test de **Résolution** : réussi si la solution optimale de (RL) est réalisable pour le problème de départ (e.g. entière)

③ Stratégie d'exploration

- priorité au noeud ayant la meilleure borne

1 Relaxations et calculs de bornes

2 Arbre de décision / Arborescence

3 Branch-and-bound

4 Règles à respecter pour un algorithme correct

5 Exemples d'applications

Application 1

$$(P) \min f(x) = 5x_1 + 7x_2 + 10x_3 + 3x_4 + x_5 \quad (10)$$

s.c.

$$x_1 - 3x_2 + 5x_3 + x_4 - 4x_5 + 2 \geq 0 \quad (11)$$

$$-2x_1 + 6x_2 - 3x_3 - 2x_4 + 2x_5 \geq 0 \quad (12)$$

$$-x_2 + 2x_3 - x_4 - 1 \geq 0 \quad (13)$$

$$x_i \in \{0, 1\} \quad i = 1, \dots, 5 \quad (14)$$

Résoudre (P) par *branch-and-bound* à l'aide des paramètres suivants :

1 Critère de séparation :

- variable de plus grand coefficient dans $f(x)$
- 2 sous-problèmes : var = 1 ou var = 0

2 Tests de Sondabilité

- TA : éval des contraintes en majorant le membre de gauche ($= G_i^k$)
- TO : éval d'une borne ($= E_k$) de la fonction-obj, comparaison avec sol si connue
- TR : si le sous-problème obtenu est trivial à résoudre

3 Stratégie d'Exploration

- PSEP : priorité au sous-problème non sondable de meilleure borne
- PSES : priorité au sous-problème "à gauche"

* Résolution par P.S.E.S. :

Mémorisation :

$$\underline{x_3=1 \quad x_2=1 \quad x_1=1 \quad x_4=0 \quad x_5=0 \quad f=22}$$

$$x_3=1 \quad x_2=1 \quad x_1=0 \quad x_4=0 \quad x_5=0 \quad f=17$$

$$\text{Min } f(x)=5x_1+7x_2+10x_3+3x_4+x_5 \quad E_0=0$$

$$x_1-3x_2+5x_3+x_4-4x_5+2 \geq 0 \quad G_1^0 = 1+5+1+2 = 9$$

$$-2x_1+6x_2-3x_3-2x_4+2x_5 \geq 0 \quad G_2^0 = 6+2 = 8$$

$$-x_2+2x_3-x_4-1 \geq 0 \quad G_3^0 = 2-1 = 1$$

$$x_i \in \{0,1\} \quad i=1,\dots,5$$

$$\text{Min } f(x)=5x_1+7x_2+3x_4+x_5+10 \quad E_1=10$$

$$x_1-3x_2+x_4-4x_5+7 \geq 0 \quad G_1^1 = 2+7 = 9$$

$$-2x_1+6x_2-2x_4+2x_5-3 \geq 0 \quad G_2^1 = 8-3 = 5$$

$$-x_2-x_4+1 \geq 0 \quad G_3^1 = 0+1 = 1$$

$$x_i \in \{0,1\}$$

$$\text{Min } f(x)=5x_1+7x_2+3x_4+x_5 \quad E_2=0$$

$$x_1-3x_2+x_4-4x_5+2 \geq 0 \quad G_1^2 = 2+2 = 4$$

$$-2x_1+6x_2-2x_4+2x_5 \geq 0 \quad G_2^2 = 8$$

$$-x_2-x_4-1 \geq 0 \quad G_3^2 = 0-1 = -1$$

$$x_i \in \{0,1\}$$

S → TA

$$\text{Min } f(x)=5x_1+3x_4+x_5+17 \quad E_3=17$$

$$x_1+x_4-4x_5+4 \geq 0 \quad G_1^3 = 2+4 = 6$$

$$-2x_1-2x_4+2x_5+3 \geq 0 \quad G_2^3 = 5$$

$$-x_4 \geq 0 \quad G_3^3 = 0$$

$$x_i \in \{0,1\}$$

$$\text{Min } f(x)=5x_1+3x_4+x_5+10 \quad E_4=10$$

$$x_1+x_4-4x_5+7 \geq 0 \quad G_1^4 = 2+7 = 9$$

$$-2x_1-2x_4+2x_5-3 \geq 0 \quad G_2^4 = 2-3 = -1$$

$$-x_4+1 \geq 0 \quad G_3^4 = 0+1 = 1$$

$$x_i \in \{0,1\}$$

S → TA

$$\text{Min } f(x)=3x_4+x_5+22 \quad E_5=22$$

$$+x_4-4x_5+5 \geq 0$$

$$-2x_4+2x_5+1 \geq 0$$

$$-x_4 \geq 0$$

$$x_i \in \{0,1\}$$

$$G_1^5 = 1+5 = 6$$

$$G_2^5 = 2+1 = 3$$

$$G_3^5 = 0$$

$$\text{Min } f(x)=3x_4+x_5+17 \quad E_6=17$$

$$+x_4-4x_5+4 \geq 0 \quad G_1^6 = 5$$

$$-2x_4+2x_5+3 \geq 0 \quad G_2^6 = 5$$

$$-x_4 \geq 0 \quad G_3^6 = 0$$

$$x_i \in \{0,1\}$$

$$\text{Min } f(x)=+x_5+25 \quad E_7=25$$

$$-4x_5+6 \geq 0 \quad G_1^7 = 6$$

$$+2x_5-1 \geq 0 \quad G_2^7 = 1$$

$$-1 \geq 0 \quad G_3^7 = -1$$

$$x_i \in \{0,1\}$$

S → TA

$$\text{Min } f(x)=+x_5+17 \quad E_{10}=17$$

$$-4x_5+4 \geq 0 \quad G_1^{10} = 4$$

$$+2x_5+3 \geq 0 \quad G_2^{10} = 5$$

$$0 \geq 0 \quad G_3^{10} = 0$$

$$x_i \in \{0,1\}$$

**S → TR
($x_5=0$)**

$$\text{Min } f(x)=+x_5+22 \quad E_8=22$$

$$-4x_5+5 \geq 0 \quad G_1^8 = 5$$

$$+2x_5+1 \geq 0 \quad G_2^8 = 3$$

$$-x_4 \geq 0 \quad G_3^8 = 0$$

$$x_i \in \{0,1\}$$

**S → TR
($x_5=0$)**

$$\text{Min } f(x)=+x_5+20 \quad E_9=20$$

$$-4x_5+5 \geq 0 \quad G_1^9 = 5$$

$$+2x_5-1 \geq 0 \quad G_2^9 = 3$$

$$-1 \geq 0 \quad G_3^9 = -1$$

$$x_i \in \{0,1\}$$

S → TA

* Résolution par P.S.E.P. :

Mémorisation :

$$x_3=1 \quad x_2=1 \quad x_1=0 \quad x_4=0 \quad x_5=0 \quad f=17$$

$$\text{Min } f(x)=5x_1+7x_2+3x_4+x_5+10 \quad E_1=10$$

$$\begin{aligned} x_1-3x_2+x_4-4x_5+7 &\geq 0 & G_1^1 &= 2+7=9 \\ -2x_1+6x_2-2x_4+2x_5-3 &\geq 0 & G_2^1 &= 8-3=5 \\ -x_2-x_4+1 &\geq 0 & G_3^1 &= 0+1=1 \\ x_i \in \{0,1\} & & & \end{aligned}$$

$$\text{Min } f(x)=5x_1+3x_4+x_5+17 \quad E_3=17$$

$$\begin{aligned} x_1+x_4-4x_5+4 &\geq 0 & G_1^3 &= 2+4=6 \\ -2x_1-2x_4+2x_5+3 &\geq 0 & G_2^3 &= 5 \\ -x_4 &\geq 0 & G_3^3 &= 0 \\ x_i \in \{0,1\} & & & \end{aligned}$$

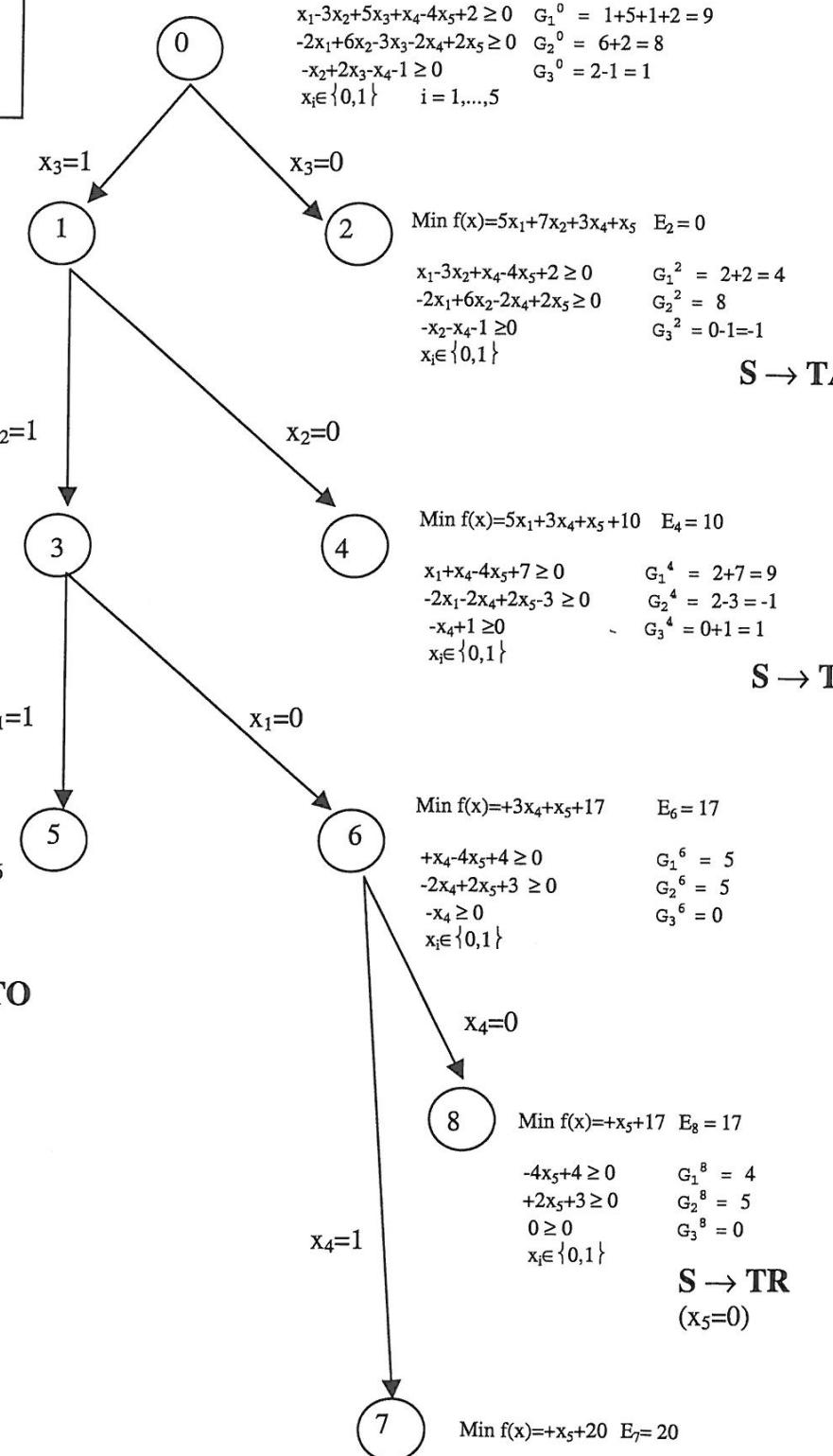
$$\text{Min } f(x)=+3x_4+x_5+22$$

$$\begin{aligned} +x_4-4x_5+5 &\geq 0 & G_1^5 &= 1+5=6 \\ -2x_4+2x_5+1 &\geq 0 & G_2^5 &= 2+1=3 \\ -x_4 &\geq 0 & G_3^5 &= 0 \\ x_i \in \{0,1\} & & & \end{aligned}$$

S → TO

$$\text{Min } f(x)=5x_1+7x_2+10x_3+3x_4+x_5 \quad E_0=0$$

$$\begin{aligned} x_1-3x_2+5x_3+x_4-4x_5+2 &\geq 0 & G_1^0 &= 1+5+1+2=9 \\ -2x_1+6x_2-3x_3-2x_4+2x_5 &\geq 0 & G_2^0 &= 6+2=8 \\ -x_2+2x_3-x_4-1 &\geq 0 & G_3^0 &= 2-1=1 \\ x_i \in \{0,1\} & \quad i=1,\dots,5 & & \end{aligned}$$



Application 2 : Méthode arborescente de Dakin

$$(P) \max f(x) = 4x_1 + 3x_2 \quad (15)$$

S.C.

$$3x_1 + 4x_2 \leq 12 \quad (16)$$

$$4x_1 + 2x_2 \leq 9 \quad (17)$$

$$x_i \in \{0, 1\} \quad i = 1, 2 \quad (18)$$

Résoudre (P) par *branch-and-bound* à l'aide des paramètres suivants :

- ① Critère de séparation :
 - variable la plus fractionnaire
- ② Tests de Sondabilité
 - TA : réussi si la relaxation linéaire (RL) n'a pas de solution admissible
 - TO : réussi si la solution de la RL est pire que la meilleure solution connue
 - TR : réussi si la solution de la RL est entière
- ③ Stratégie d'Exploration
 - priorité au sous-problème non sondable de meilleure borne

Arborescence résultante visible sur la figure 2.4 en page 41 du livre de C. Prins

Résumé de ce qui a été vu

Branch-and-Bound

- Règles de Séparation
- Tests de sondabilité (TA, TO, TR)
- Stratégies d'exploration
- Exemples d'application

Pour aller plus loin

- Complexité
 - Comment identifier la complexité d'un problème
 - Méthodes de réduction polynomiale
- Résolution de problèmes polynomiaux
- Inégalités valides et méthodes de coupes

Programmation Dynamique

Sandra U. Ngueveu

Toulouse INP N7 - LAAS/CNRS
<https://homepages.laas.fr/sungueve>

<https://moodle-n7.inp-toulouse.fr/course/view.php?id=2380>

2A SN B-M-L : 2024/2025

Sandra U. Ngueveu (N7 - LAAS)

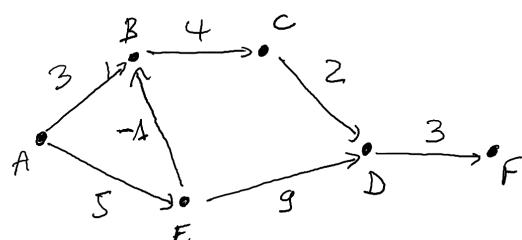
R.O. - Programmation Dynamique - 2A SN

2A SN B-M-L : 2024/2025

1 / 7

Application au calcul de plus court chemin (Bellman-Ford)

Calcul des plus court chemins du sommet A aux autres sommets du graphe



- f_i^k = coût du plus court chemin de A à i contenant au plus k arcs
- relation de récurrence : $f_i^k = \min_{\forall j \in \text{Pred}(i)} f_j^{k-1} + c_{ji}$

itération	A	B	C	D	E	F
0	0	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$
1	0	3^A	$+\infty$	$+\infty$	$+\infty$	$+\infty$
2	0	3^A	7^B	14^E	5^A	$+\infty$
3	0	3^A	7^B	9^C	5^A	17^D
4	0	3^A	7^B	9^C	5^A	12^D
5	0	3^A	7^B	9^C	5^A	12^D

Complexité : $O(n^2)$

Sandra U. Ngueveu (N7 - LAAS)

R.O. - Programmation Dynamique - 2A SN

2A SN B-M-L : 2024/2025

2 / 7

Généralités

Principe

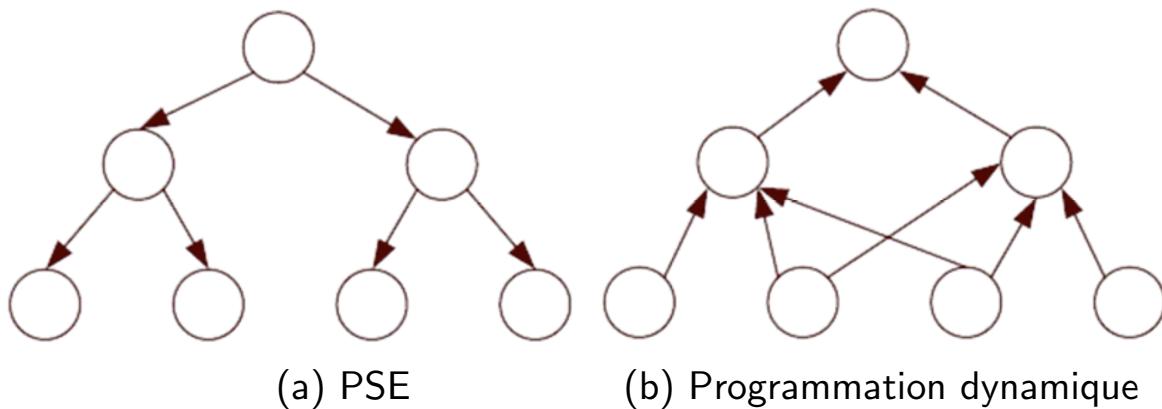
Pour résoudre un problème, commencer par résoudre des problèmes de taille inférieure et utiliser ces valeurs pour calculer la solution de problèmes de plus en plus grands, jusqu'à obtenir la solution du problème initial.

En pratique, la programmation dynamique

- décompose le problème en étapes
- chaque étape corresponds à un sous-problème résolu **optimalement** compte tenu des infos obtenues lors des étapes précédentes
- nécessite de trouver une relation de récurrence entre les valeurs des critères de deux niveaux successifs
- parcourt à rebours (de la dernière décision aux précédentes) le processus de décision séquentiel associé au problème

Programmation dynamique vs *Branch-and-Bound*

Similitudes : les deux méthodes garantissent l'**optimalité** de la solution obtenue tout en essayant de contenir l'**explosion combinatoire**.



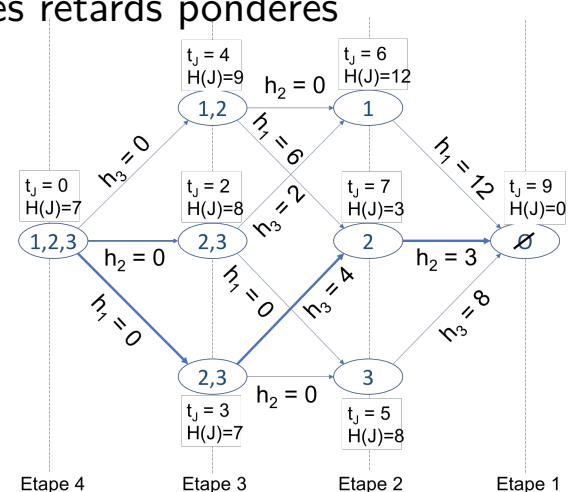
Différences :

- ordre d'exploration : "du haut vers le bas" vs "du bas vers le haut"
- en prog. dyn. classique chaque niveau ne peut être entamé qu'après avoir finalisé les niveaux qui précédent.
- pour appliquer la prog. dyn., il faut avoir démontré que la solution optimale à un problème est composée de solutions optimales à des sous-problèmes

Application en ordonnancement de tâches

Problème : Minimisation de la somme des retards pondérés

tâche i	1	2	3
durée p_i	3	2	4
poids w_i	3	1	2
date échue d_i	5	6	5



- c_j = date de fin de la tâche j
- J = ensemble (non ordonné) de tâches placées en dernière position
- $H(J)$ = coût d'un sommet = retard pondéré minimal
- $H(J) = \sum_{j \in J} h_j(c_j)$ avec $h_j(c_j) = w_j T_j = w_j(\max\{0, c_j - d_j\})$
- relation de récurrence : $H(J) = \min_{j \in J} \{h_j(t_j + p_j) + H(J \setminus \{j\})\}$

Complexité : $O(2^n)$

Application au problème de rendu de monnaie

Problème : Soit un montant M et des pièces de valeurs v_1, v_2, \dots, v_n . Trouver des entiers x_i tels que

$$\min \sum_{i=1}^n x_i \quad \text{s.c.} \quad \sum_{i=1}^n v_i x_i = M$$

- $C_{i,j}$ = nombre minimum de pièces pour produire exactement le montant j en utilisant seulement les pièces de valeurs v_1, \dots, v_i
- initialisation : $C_{i,0} = 0, \forall i$
- relation de récurrence : $C_{i,j} = \min C_{i-1,j}, 1 + C_{i,j-v_i}$

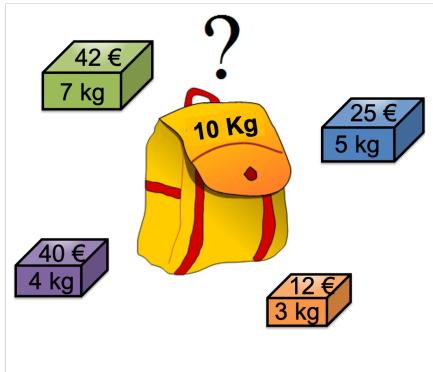
Illustration avec $M = 8$ et des pièces de valeurs $v_1 = 1, v_2 = 4$ et $v_3 = 6$

	0	1	2	3	4	5	6	7	8
$v_1 = 1$	0	1	2	3	4	5	6	7	8
$v_2 = 4$	0	1	2	3	1	2	3	4	2
$v_3 = 6$	0	1	2	3	1	2	1	2	2

Complexité : $O(nM)$

Application au problème du sac à dos

Problème :



$$\max \sum_{i=1}^n v_i x_i \quad (1)$$

$$\text{s.c. } \sum_{i=1}^n w_i x_i \leq Q \quad (2)$$

$$x_i \in \{0, 1\}, \forall i \in \{1, \dots, n\} \quad (3)$$

- $C_{i,j}$ = valeur maximale des objets que l'on peut transporter si le poids maximal permis est j et que les objets que l'on peut inclure sont ceux numérotés de 1 à i
- initialisation : $C_{i,0} = 0, \forall i$
- relation de récurrence : $C_{i,j} = \max\{C_{i-1,j}, C_{i-1,j-w_i} + c_i\}$

Complexité : $O(nQ)$